ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО 10303-21— 2022

Системы автоматизации производства и их интеграция

ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ

Часть 21

Методы реализации. Кодирование открытым текстом структуры обмена

(ISO 10303-21:2016, IDT)

Издание официальное

Москва
Российский институт стандартизации
2022

gost.e-ecolog.ru

Предисловие

1 ПОДГОТОВЛЕН Федеральным государственным бюджетным учреждением «Российский институт стандартизации» (ФГБУ «Институт стандартизации») на основе собственного перевода на русский язык англоязычной версии стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 194 «Кибер-физические системы»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 17 ноября 2022 г. № 1301-ст

4 Настоящий стандарт идентичен международному стандарту ИСО 10303-21:2016 «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 21. Методы реализации. Кодирование открытым текстом структуры обмена» (ISO 10303-21:2016 «Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure», IDT).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты, сведения о которых приведены в дополнительном приложении ДА

5 ВЗАМЕН ГОСТ Р ИСО 10303-21—2002

*Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (*gost.e-ecolog.ru

© ISO, 2016
© Оформление. ФГБУ «Институт стандартизации», 2022

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения 1
2 Нормативные ссылки 1
3 Термины, определения и сокращения 2
3.1 Термины и определения 2
3.2 Сокращения 3
4 Основополагающие концепции и допущения структуры обмена 4
4.1 Общие положения 4
4.2 Нотационные и типографские соглашения 4
4.3 Соответствие 4
5 Формальные определения 5
5.1 Формальная нотация 5
5.2 Определение основного алфавита 5
5.3 Структура обмена 6
5.4 Определение лексем 6
5.5 Структура обмена в WSN 7
5.6 Разделители лексем 8
6 Лексемы 9
6.1 Типы лексем 9
6.2 Специальные лексемы 9
6.3 Ключевые слова 9
6.4 Кодирование простых типов данных 10
6.5 Кодирование секций привязки, ссылочной секции и подписи 16
7 Структурированные типы данных 18
7.1 Параметр LIST (список) 18
7.2 Список элементов привязки 18
8 Заголовочная секция 19
8.1 Структура заголовочной секции 19
8.2 Декларации заголовочной секции 19
8.3 Объекты заголовочной секции, определенные пользователем 26
9 Секция привязки 26
9.1 Структура секции привязки 26
9.2 Элемент привязки 27
10 Ссылочная секция 29
10.1 Структура ссылочной секции 29
10.2 Ссылка на URI 30
11 Секции данных 32
11.1 Структура секции данных 32
11.2 Экземпляры объектов секции данных 32
11.3 Экземпляры объектов секции данных, определяемые пользователем 33
12 Отображение из EXPRESS в структуру обмена 33
12.1 Отображение типов данных EXPRESS 33
12.2 Отображение типов данных объекта из языка EXPRESS 42
12.3 Отображение элемента EXPRESS для SCHEMA 52
12.4 Отображение элемента EXPRESS для CONSTANT 52
12.5 Отображение элемента EXPRESS для RULE 52
12.6 Комментарии 52

13 Печатное представление структур обмена 52
14 Секции подписи 53
14.1 Структура секции подписи 53
Приложение A (обязательное) Представление файла на носителе данных 54
Приложение B (обязательное) Соглашения по записи в синтаксической нотации Вирта 57
Приложение C (обязательное) Регистрация информационного объекта 58
Приложение D (обязательное) Форма заявки о соответствии реализации протоколу 59
Приложение E (обязательное) Множество EXPRESS-схем в структуре обмена 61
Приложение F (обязательное) Привязка ECMAScript к секции привязки 66
Приложение G (обязательное) Отображение UUID в имена элементов привязки 72
Приложение H (справочное) Пример полной структуры обмена 74
Приложение I (справочное) Пример распределенной структуры обмена 75
Приложение J (справочное) Примеры констант EXPRESS для определения единиц измерения 77
Приложение K (справочное) Рекомендуемые имена типов файлов 78
Приложение L (справочное) Руководство по распечатке структуры обмена 79
Приложение M (справочное) Журнал изменений 80
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов национальным стандартам 81
Библиография 82

Введение

Стандарты серии ИСО 10303 распространяются на компьютерное представление информации об изделиях и обмен данными об изделиях. Их целью является обеспечение нейтрального механизма, способного описывать изделия на всем протяжении их жизненного цикла. Этот механизм применим не только для обмена файлами в нейтральном формате, но является также основой для реализации и совместного доступа к базам данных об изделиях и организации архивирования.

Настоящий стандарт устанавливает механизм, который позволяет представлять данные об изделии для передачи из одной вычислительной системы в другую, используя язык EXPRESS, описанный в ИСО 10303-11.

Основные разделы настоящего стандарта:

- определение синтаксиса структуры обмена:
- преобразование из EXPRESS-схемы в заданный синтаксис.

Примечание — Примеры использования EXPRESS в настоящем стандарте не соответствуют каким-либо правилам стиля. Напротив, иногда в примерах используют искаженный стиль, чтобы сохранить место или сконцентрироваться на важных вопросах. Примеры не ставят целью отразить содержание информационных моделей, определенных в других стандартах серии ИСО 10303. Данные примеры предназначены для показа конкретных особенностей EXPRESS или структуры обмена. Многие примеры даны с аннотациями, не согласующимися с синтаксическими правилами настоящего стандарта. Такие аннотации введены символическими стрелками: или горизонтальными '-->' или вертикальными. При составлении правил просмотра текста эти аннотации должны быть игнорированы. Должно быть также игнорировано любое сходство между примерами и нормативными моделями, определенными в других стандартах серии ИСО 10303. В настоящем стандарте приведены несколько примеров отображения. В некоторые примеры для улучшения читаемости вставлены дополнительные пробелы и новые строки. Эти пробелы и новые строки не должны появляться в структуре обмена.

Настоящее издание включает следующие технические изменения к ИСО 10303-21:2002:

- разделы привязки и ссылки позволяют определять объекты и значения во внешних файлах;
- EXPRESS-схема **schema_population** управляет тем, как экземпляры распределяются по нескольким файлам;
- структура обмена может быть сжата и сохранена в архиве;
- строки могут содержать символы UTF-8 кодированных октетов вместо \X2\ и \X4\ управляющих директив;

структуре обмена может быть предоставлена одна или несколько сигнатур для проверки ее содержания и проверки полномочий автора;

структура обмена может ссылаться на значения и экземпляры, определенные в ее EXPRESS-схеме;

- структура обмена может включать ECMAScript;
- исходный документ представлен в HTML.

Примечание — Приложения могут использовать новую версию для обеспечения более широкого совместного использования данных, сокращения объема памяти, сокращения затрат на разработку, повышения производительности и безопасности данных.

Все структуры обмена, которые кодируются в соответствии с предыдущим изданием ИСО 10303-21, также соответствуют этому изданию.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы автоматизации производства и их интеграция

ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ

Часть 21

Методы реализации.
Кодирование открытым текстом структуры обмена

Industrial automation systems and integration. Product data representation and exchange. Part 21. Implementation methods. Clear text encoding of the exchange structure

Дата введения — 2023—01—01

1 Область применения

Настоящий стандарт устанавливает формат структуры обмена, использующий кодирование открытым текстом данных об изделии, для которого концептуальная модель определена в языке EXPRESS (ИСО 10303-11). Формат обмена пригоден для передачи данных об изделии между вычислительными системами и для распределения данных об изделии между несколькими вычислительными системами.

Определено преобразование из языка EXPRESS в синтаксис структуры обмена. В синтаксис структуры обмена может быть преобразована любая EXPRESS-схема.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты [для датированных ссылок применяют только указанное издание ссылочного стандарта, для недатированных — последнее издание (включая все изменения)]:

ISO 639-2, Codes for the representation of names of languages — Part 2: Alpha-3 code.(Коды для представления названий языков. Часть 2. Код ALPHA-3)

ISO 8601:2004[1)], Data elements and interchange formats — Information interchange — Representation of dates and times (Элементы данных и форматы обмена информацией. Обмен информацией. Представление дат и времени)

ISO 10303-1:1994[2)], Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles (Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1. Общие представления и основополагающие принципы)

ISO 10303-11:2004, Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual (Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 11. Методы описания. Справочное руководство по языку EXPRESS)

[1)] Заменен на ISO 8601-1:2019; ISO 8601-2:2019. Однако для однозначного соблюдения требования настоящего стандарта, выраженного в датированной ссылке, рекомендуется использовать только указанное в этой ссылке издание.

[2)] Заменен на ISO 10303-1:2021. Однако для однозначного соблюдения требования настоящего стандарта, выраженного в датированной ссылке, рекомендуется использовать только указанное в этой ссылке издание.

Издание официальное

ISO/IEC 8824-1, Information technology — Abstract Syntax Notation One (ASN.1) — Part 1: Specification of basic notation [Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации]

ISO/IEC 8859-1, Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1 (Информационные технологии. 8-битовые однобайтовые наборы кодированных графических знаков. Часть 1. Латинский алфавит № 1)

ISO/IEC 8859 (all parts), Information technology — 8-bit single-byte coded graphic character sets (Информационная технология. 8-битовые однобайтовые наборы кодированных графических знаков)

ISO/IEC 10646, Information technology — Universal Coded Character Set (UCS) (Информационная технология. Универсальный набор кодированных символов (UCS))

ISO/IEC 16262[1)], Information technology — Programming languages, their environments and system software interfaces — ECMAScript language specification (Информационные технологии. Языки программирования, их среды и системные программные интерфейсы. Спецификация языка ECMAScript)

ISO/IEC 21320-1, Information technology — Document Container File — Part 1: Core (Информационная технология. Файл контейнера документа. Часть 1. Ядро)

3 Термины, определения и сокращения

3.1 Термины и определения

3.1.1 Термины, определенные в ИСО/МЭК 8859-1

В настоящем стандарте применен термин **байт** (byte).

3.1.2 Термины, определенные в ИСО/МЭК 10646-1

В настоящем стандарте применены следующие термины:

базовая многоязыковая плоскость (basic multilingual plane);

символ (character);

кодовая точка (code point);

схема кодирования (encoding scheme);

графический символ (graphic character);

октет (octet).

3.1.3 Термины, определенные в ИСО 10303-1

В настоящем стандарте применены следующие термины:

- **прикладной протокол;** ПП (application protocol; AP);
- **структура обмена** (exchange structure).

3.1.4 Термины, определенные в ИСО 10303-11

В настоящем стандарте применены следующие термины:

- **экземпляр сложного объекта** (complex entity instance);
- **тип данных** (data type);
- **константа EXPRESS** (EXPRESS constant);
- **объект** (entity);
- **частное значение сложного объекта** (partial complex entity instance);
- **экземпляр простого объекта** (simple entity instance);
- **лексема** (token).

3.1.5 Термины, определенные в ИСО/МЭК 16262

В настоящем стандарте применен термин **скрипт ECMA** (ECMAScript).

3.1.6 Термины, определенные в ИСО/МЭК 21320-1

В настоящем стандарте применен термин **архив ZIP** (Archive ZIP).

3.1.7 Другие термины

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1.7.1 **URI** (Uniform Resource Identifier): Компактная строка символов для обозначения абстрактного или физического ресурса.

[Источник: IETF RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax]

[1)] Заменен на ISO/IEC 22275:2018.

3.1.7.2 **обозначение фрагмента URI** (URI Fragment Identifier): Дополнительная ссылка, которая используется, когда доступ к ресурсу URI успешно выполнен.

[Источник: Fragment identifier in IETF RFC 2396]

3.1.7.3 **запросная составляющая URI** (URI Query component): Строка информации, интерпретируемая ресурсом URI.

[Источник: Query component in IETF RFC 2396]

3.1.7.4 **универсальный уникальный идентификатор**; UUID (Universally Unique Identifier, UUID): Иидентификатор, уникальный как в пространстве, так и во времени.

[Источник: IETF RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace]

3.1.7.5 **код Base64** (Base64 encoding): Символ, кодированный в системе CMS.

[Источник: IETF RFC 4648: The Base16, Base32 and Base64 Data Encodings]

3.1.7.6 **синтаксис криптографических сообщений;** CMS (Cryptographic Message Syntax, CMS): Спецификация для формы и содержания цифровой подписи.

[Источник: IETF RFC 5652: Cryptographic Message Syntax (CMS)]

3.1.7.7 **дайджест сообщений** (message digest): Хэшированное значение, вычисленное по содержимому и включенное в данные подписи.

[Источник: IETF RFC 5652 Section 5]

3.1.7.8 **привязка** (anchor): Обозначение, на которое может быть дана ссылка.

3.1.7.9 **основной алфавит** (basic alphabet): Набор символов UCS соответствующих кодовым точкам от U+0020 до U+007E и от U+0080 до U+10FFFF.

3.1.7.10 **кодирование открытым текстом** (clear text encoding): Кодирование информации с использованием последовательности кодов для символов основного алфавита.

3.1.7.11 **директива управления** (control directive): Последовательность символов в основном алфавите.

3.1.7.12 **ключевое слово** (keyword): Особая последовательность символов, обозначающая объект или определенный тип в структуре обмена.

3.1.7.13 **ссылка** (reference): Адрес точки привязки в обменной структуре или в ресурсе иного типа, который может быть преобразован в обменную структуру.

3.1.7.14 **ресурс** (resource): Физическая или абстрактная структура обмена, на которую указывает URI.

3.1.7.15 **секция** (section): Набор данных одной и той же категории информации.

3.1.7.16 **последовательный файл** (sequential file): Файл, который может быть доступен только последовательным способом.

3.1.7.17 **подпись** (signature): Криптографически закодированная аутентификация содержимого.

3.1.7.18 метка (tag): Дополнительная информация о точке привязки, которая находится вне области действия схемы.

3.1.7.19 **разделитель лексем** (token separator): Последовательность из одного или нескольких символов из основного алфавита, которая разделяет любые две лексемы.

3.2 Сокращения

В настоящем стандарте применены следующие сокращения:

BMP — основной многоязычный уровень (Basic Multilingual Plane, BMP);

CMS — синтаксис криптографических сообщений (Cryptographic Message Syntax);

IETF — рабочая группа инженеров Интернета (Internet Engineering Task Force);

GUID — глобальный уникальный идентификатор (Globally Unique Identifier);

OID — идентификатор объекта (Object Identifier);

SDAI — стандартный интерфейс доступа к данным (standard data access interface);

UCS — универсальный набор символов (Universal Character Set);

UTF — кодировка UTF (Universal Character Set + Transformation Format);

URI — универсальный идентификатор ресурса (Uniform Resource Identifier);

UUID — универсальный уникальный идентификатор (Universally Unique Identifier);

WSN — синтаксическая нотация Вирта (Wirth Syntax Notation).

4 Основополагающие концепции и допущения структуры обмена

4.1 Общие положения

Для того чтобы облегчить синтаксический анализ с помощью программных средств, структура обмена описана однозначной, контекстно-свободной грамматикой. Грамматика выражена в синтаксической нотации Вирта, которая описана в приложении В. Представление данных об изделии в структуре обмена определяется с использованием отображения из языка EXPRESS в синтаксис структуры обмена.

4.2 Нотационные и типографские соглашения

Любые кавычки, используемые в настоящем стандарте, не являются частью текста структуры обмена, но служат для отделения этого текста. Это положение применимо ко всем местам в тексте, где используются кавычки. Таблицы 2—4 определяют исключения из этого правила, так как кавычки, используемые в этих таблицах, составляют часть правил WSN.

Стандартами серии ИСО 8859 каждому символу присваивается обозначающее имя. Когда это имя используют в настоящем стандарте, для отличия от обычного текста, оно выделено курсивом. Так, *запятую* (comma) используют для ссылки на ",", *подчеркивание* (low line) — для ссылки на "_", а *прописную букву A* (capital letter A) — для ссылки на "A".

В примерах настоящего стандарта, где требуются пояснения, они введены последовательностью –––>.

4.3 Соответствие

Установлены два уровня соответствия:

- синтаксическое соответствие структуры обмена: структура обмена соответствует настоящему стандарту, если удовлетворены требования этого стандарта;
- схематическое соответствие структуры обмена: экземпляры, представленные в структуре обмена, соответствуют схемам, указанным в заголовочной секции соответствующей структуры обмена, если удовлетворены все требования и ограничения данных схем и требования отображения каждого экземпляра или группы экземпляров, установленные в разделах 11 и 12.

Примечание — В приложении Е определены методы оценки соответствия схемы, когда структура обмена содержит ряд секций данных, связанных с различными EXPRESS-схемами.

Синтаксическое соответствие является необходимым условием для схематического соответствия.

Настоящим стандартом установлены три класса синтаксического соответствия:

- структура обмена будет синтаксически соответствовать классу 3, если она определяет экземпляры значений, если она ссылается на константы EXPRESS или если она определяет функциональные возможности, используя скрипты ECMAScript;
- иначе структура обмена будет соответствовать классу 2, если она имеет секцию ссылок или использует многофайловую структуру архивов ZIP;
- иначе структура обмена будет синтаксически соответствовать классу 1.

Примечания

1 Процессор в случае предыдущего издания настоящего стандарта может осуществлять разбор файлов класса соответствия 1, игнорируя секции привязки и подписи, и может разбирать файлы класса соответствия 2, используя препроцессор для удовлетворения ссылок на внешние объекты в секции ссылок.

2 Класс синтаксического соответствия обменной структуры обозначается значением атрибута **implementation_level** (уровень реализации), представляющего описание файла объекта **file_description** в заголовочной секции (см. 8.2.2). Если атрибут имеет значение "4;1", то эта обменная структура имеет класс синтаксического соответствия 1. Если атрибут имеет значение "4;2", то эта обменная структура имеет класс синтаксического соответствия 2. Если атрибут имеет значение "4;3", то эта обменная структура имеет класс синтаксического соответствия 3.

Реализация, претендующая на схематическое соответствие настоящему стандарту, должна читать или (и) записывать файлы, которые демонстрируют схематическое соответствие наряду с синтаксическим соответствием.

5 Формальные определения

5.1 Формальная нотация

В настоящем стандарте для определения синтаксиса структуры обмена использована синтаксическая нотация Вирта (WSN), описанная в приложении В.

5.2 Определение основного алфавита

Алфавит структуры обмена определен как кодовые точки от U+0020 до U+007E и от U+0080 до U+10FFFF ИСО/МЭК 10646. Этот алфавит представлен в структуре обмена, используя схему кодирования UTF-8 по ИСО/МЭК 10646. Таблица 1 делит основной алфавит на подмножества.

Результатом схемы кодирования UTF-8 является единичный октет, имеющий шестнадцатеричное значение от 20 до 7E для каждого символа LATIN_CODEPOINT (кодовые точки латинского алфавита) и последовательность октетов с шестнадцатеричными значениями от 80 до F4 для каждого символа HIGH_CODEPOINT (верхние кодовые точки). Октеты со значениями вне этих диапазонов должны при обработке обменной структуры игнорироваться.

Примечание — Набор символов LATIN_CODEPOINT (кодовые точки латинского алфавита) эквивалентен базовому алфавиту из первого и второго изданий ИСО 10303-21. Представление UTF-8 кодовых точек от U+0020 до U+007E то же самое, что и символы от G(02/00) до G(07/14) в ИСО/МЭК 8859-1, которые определяют базовый алфавит в ранних изданиях. Если требуется совместимость с предыдущими изданиями ИСО 10303-21, можно избежать использования кодовых точек HIGH_CODEPOINT (верхние кодовые точки).

Таблица 1 — Определение подмножеств основного алфавита по WSN

```
SPACE     = " " .

DIGIT     = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7"
          | "8" | "9" .

LOWER     = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h"
          | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p"
          | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x"
          | "y" | "z" .

UPPER     = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H"
          | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P"
          | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X"
          | "Y" | "Z" | "_" .

SPECIAL   = "!" | """" | "*" | "$" | "%" | "&" | "." | "#"
          | "+" | "," | "-" | "(" | ")" | "?" | "/" | ":"
          | ";" | "<" | "=" | ">" | "@" | "[" | "]" | "{"
          | "|" | "}" | "^" | "`" | "~" .

REVERSE_SOLIDUS  = "\" .

APOSTROPHE = "'" .

LATIN_CODEPOINT = SPACE | DIGIT | LOWER | UPPER | SPECIAL
          | REVERSE_SOLIDUS | APOSTROPHE

HIGH_CODEPOINT = (от U+0080 до U+10FFFF, см. 5.2)
```

5.3 Структура обмена

Структура обмена должна быть представлена в виде последовательного файла с использованием кодирования открытым текстом. Структура обмена должна содержать заголовочную секцию и четыре необязательных секций данных: секция привязки, ссылочная секция, одна или более секций данных и одна или более секций подписи. Роль каждой секции описывается ниже в том же порядке, в котором они отображаются в структуре обмена.

Заголовочная секция представляет данные, относящиеся к самой структуре обмена. Структура заголовочной секции определена в разделе 8.

В секции привязки предоставлены внешние имена для объектов и значений, на которые возможны ссылки из других структур обмена. Определение структуры секции привязки содержится в разделе 9.

В секции ссылок предоставлены ссылки на объекты и значения, которые определены во внешних структурах обмена. Определение структуры секции ссылок содержится в разделе 10.

Секция данных представляет данные, которые должны быть переданы. Структура секции данных определена в разделе 11.

Секция подписи обеспечивает достоверность передаваемых данных и подтверждает источник данных. Определение структуры секции подписи содержится в разделе 14.

Структура обмена определена с помощью WSN в таблице 3.

Примечание — Заголовочная секция располагается в начале файла, поскольку в ней содержится определение контекстной информации для оставшейся части структуры обмена. Секция привязки и секция ссылок появляются следующими, поскольку они определяют то, как структура связана с другими файлами. Размещение этих секций возле начала файла позволяет поисковым системам находить эти зависимости без чтения структуры в целом. Секция подписи располагается в конце файла таким образом, чтобы можно было добавлять новые подписи без нарушения текста, который утвержден предыдущими подписями.

Структура обмена является потоком 8-битных байтов, которые кодируются графическими символами основного алфавита. Графические символы группируются в распознаваемые последовательности, называемые лексемами. Лексемы могут быть отделены разделителями. Структуру обмена можно рассматривать как последовательность лексем и их разделителей.

Структура обмена может быть сжата и сохранена в архиве с использованием организации данных, описанной в А.4, приложение А.

5.4 Определение лексем

Лексемы, используемые в структуре обмена, определены средствами WSN в таблице 2.

Таблица 2 — Определение лексем средствами WSN

```
KEYWORD             = USER_DEFINED_KEYWORD | STANDARD_KEYWORD .

USER_DEFINED_KEYWORD = "!" UPPER { UPPER | DIGIT } .

STANDARD_KEYWORD    = UPPER { UPPER | DIGIT } .

SIGN                = "+" | "-" .

INTEGER             = [ SIGN ] DIGIT { DIGIT } .

REAL                = [ SIGN ] DIGIT { DIGIT } "." { DIGIT }
                      [ "E" [ SIGN ] DIGIT { DIGIT } ] .

STRING              = "'" { SPECIAL | DIGIT | SPACE | LOWER | UPPER |
                      HIGH_CODEPOINT |
                      APOSTROPHE APOSTROPHE |
                      REVERSE_SOLIDUS REVERSE_SOLIDUS |
                      CONTROL_DIRECTIVE } "'" .

ENTITY_INSTANCE_NAME    = "#" ( DIGIT ) { DIGIT } .

VALUE_INSTANCE_NAME       = "@" ( DIGIT ) { DIGIT } .

CONSTANT_ENTITY_NAME      = "#" ( UPPER ) { UPPER | DIGIT } .

CONSTANT_VALUE_NAME       = "@" ( UPPER ) { UPPER | DIGIT } .

LHS_OCCURRENCE_NAME       = ( ENTITY_INSTANCE_NAME | VALUE_INSTANCE_NAME ) .

RHS_OCCURRENCE_NAME       = ( ENTITY_INSTANCE_NAME | VALUE_INSTANCE_NAME |
                              CONSTANT_ENTITY_NAME | CONSTANT_VALUE_NAME) .

ANCHOR_NAME         = "<" URI_FRAGMENT_IDENTIFIER ">" .

TAG_NAME            = ( UPPER | LOWER) { UPPER | LOWER | DIGIT } .

RESOURCE            = "<" UNIVERSAL_RESOURCE_IDENTIFIER ">" .

ENUMERATION         = "." UPPER { UPPER | DIGIT } "." .

HEX                 = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                      "8" | "9" | "A" | "B" | "C" | "D" | "E" | "F" .

BINARY              = """ ( "0" | "1" | "2" | "3" ) { HEX } """ .

SIGNATURE_CONTENT = BASE64 .
```

5.5 Структура обмена в WSN

Синтаксис структуры обмена установлен в таблице 3. Таблица 3 ссылается на лексемы, определенные в таблице 2. Отношение между синтаксисом и EXPRESS-схемой установлено в разделе 12.

Таблица 3 — Структура обмена в WSN

```
EXCHANGE_FILE      = "ISO-10303-21;"
                     HEADER_SECTION [ ANCHOR_SECTION ]
                     [ REFERENCE_SECTION ] { DATA_SECTION }
                     "END-ISO-10303-21;" { SIGNATURE_SECTION }.

HEADER_SECTION     = "HEADER;"
                     HEADER_ENTITY HEADER_ENTITY HEADER_ENTITY
                     [HEADER_ENTITY_LIST]
                     "ENDSEC;" .
HEADER_ENTITY_LIST = HEADER_ENTITY { HEADER_ENTITY } .
HEADER_ENTITY      = KEYWORD  "(" [ PARAMETER_LIST ] ")" ";" .

PARAMETER_LIST     = PARAMETER { "," PARAMETER } .
PARAMETER          = TYPED_PARAMETER  |
                     UNTYPED_PARAMETER | OMITTED_PARAMETER  .
TYPED_PARAMETER    = KEYWORD "(" PARAMETER ")" .
UNTYPED_PARAMETER  = "$" | INTEGER | REAL | STRING | RHS_OCCURENCE_NAME
                     | ENUMERATION | BINARY | LIST .
OMITTED_PARAMETER  = "*" .
LIST               = "(" [ PARAMETER { "," PARAMETER } ] ")" .

ANCHOR_SECTION     = "ANCHOR;" ANCHOR_LIST "ENDSEC;" .
ANCHOR_LIST        = { ANCHOR } .
ANCHOR             = ANCHOR_NAME "=" ANCHOR_ITEM { ANCHOR_TAG } ";" .
ANCHOR_ITEM        = "$" | INTEGER | REAL | STRING | ENUMERATION | BINARY
                     | RHS_OCCURRENCE_NAME | RESOURCE | ANCHOR_ITEM_LIST .
ANCHOR_ITEM_LIST   = "(" [ ANCHOR_ITEM { "," ANCHOR_ITEM } ] ")" .
ANCHOR_TAG         = "{" TAG_NAME ":" ANCHOR_ITEM "}" .

REFERENCE_SECTION  = "REFERENCE;" REFERENCE_LIST "ENDSEC;" .
REFERENCE_LIST     = { REFERENCE } .
REFERENCE          = LHS_OCCURRENCE_NAME "=" RESOURCE ";" .

DATA_SECTION       = "DATA" [ "(" PARAMETER_LIST ")" ] ";"
                     ENTITY_INSTANCE_LIST "ENDSEC;" .
ENTITY_INSTANCE_LIST = { ENTITY_INSTANCE } .
ENTITY_INSTANCE    = SIMPLE_ENTITY_INSTANCE | COMPLEX_ENTITY_INSTANCE .
SIMPLE_ENTITY_INSTANCE  = ENTITY_INSTANCE_NAME "=" SIMPLE_RECORD ";" .
COMPLEX_ENTITY_INSTANCE = ENTITY_INSTANCE_NAME "=" SUBSUPER_RECORD ";" .
SIMPLE_RECORD      = KEYWORD "(" [ PARAMETER_LIST ] ")" .
SUBSUPER_RECORD    = "(" SIMPLE_RECORD_LIST ")" .
SIMPLE_RECORD_LIST = SIMPLE_RECORD { SIMPLE_RECORD } .

SIGNATURE_SECTION  = "SIGNATURE" SIGNATURE_CONTENT "ENDSEC;".
```

5.6 Разделители лексем

Разделитель лексемы является элементом, отделяющим две лексемы. Разделителями являются пробел, явные директивы управления печатью и комментарии. Разделитель может появиться между терминальными или нетерминальными лексемами таблицы 3. В том месте, где может появиться один разделитель, может появиться любое число разделителей. Разделитель не должен появляться внутри лексем, за исключением того, что явные директивы управления печатью могут появляться внутри чисел в двоичном представлении и внутри строк. Директивы управления печатью определены в разделе 13.

Примечание — *Пробел* — это единственный символ пробела, разделяющий лексемы. Разделители строк, такие как *перевод строки* или *возврат каретки*, и другие управляющие символы, такие как *перевод стра-*

ницы или *символ табуляции*, могут быть использованы в структуре обмена, но в соответствии с 5.2 они должны быть проигнорированы при обработке структуры обмена. Поэтому разделители строк могут присутствовать в структуре обмена, включая разделители лексем.

Комментарий должен быть закодирован как *косая черта, звездочка* "/*", за которым следует любое число символов из основного алфавита, и завершаться *звездочка, косая черта* "*/". Любое появление комбинации "/*" после первого появления не имеет значения, т. е. комментарии не могут быть вложены. Все графические символы, появляющиеся внутри комментария, не имеют значения для структуры обмена и предназначены только для чтения людьми.

6 Лексемы

6.1 Типы лексем

В структуре обмена лексема является специальной лексемой, ключевым словом, кодированием простого типа данных или кодированием IETF.

6.2 Специальные лексемы

Для открытия структуры обмена должна быть использована специальная лексема "ISO-10303-21;", а для закрытия структуры обмена — "END-ISO-I0303-21;".

Для того чтобы открыть или закрыть заголовочную секцию структуры обмена, должны быть использованы специальные лексемы "HEADER;" или "ENDSEC;" соответственно.

Для того чтобы открыть необязательную секцию привязки структуры обмена, должна быть использована специальная лексема "ANCHOR;", а для того чтобы закрыть секцию привязки структуры обмена, должна быть использована специальная лексема "ENDSEC;".

Для того чтобы открыть необязательную ссылочную секцию структуры обмена, должна быть использована специальная лексема "REFERENCE;", а для того чтобы закрыть ссылочную секцию структуры обмена, должна быть использована специальная лексема "ENDSEC;".

Для того чтобы открыть или закрыть секцию данных структуры обмена, должны быть использованы специальные лексемы "DATA" или "ENDSEC;" соответственно.

Для того чтобы открыть необязательную секцию подписи структуры обмена, должна быть использована специальная лексема "SIGNATURE", а для того чтобы закрыть секцию подписи структуры обмена, должна быть использована специальная лексема "ENDSEC;".

Для представления объекта, значение которого не указано в структуре обмена, используется специальный *знак доллара* (лексема "$").

Специальную лексему *звездочка* "*" используют для представления предмета, значение которого не представлено в структуре обмена, но может быть выведено из других величин в соответствии с правилами, приведенными в EXPRESS-схеме (см. 12.2.6).

Специальные лексемы *точка с запятой* (";"), круглые *скобки* ("(", ")"), *запятая* (",") и *косая черта* ("/") используют как знаки препинания в структуре обмена.

6.3 Ключевые слова

Ключевые слова являются последовательностями графических символов, указывающими объект или определенный тип в структуре обмена. Ключевые слова должны состоять из *прописных букв*, *цифр*, *подчеркивания* и, возможно, *восклицательного знака* "!" *Восклицательный знак* должен появляться не более одного раза и только как первый символ в ключевом слове.

Ключевые слова могут быть таковыми, определенными в схеме или заданными пользователем. Ключевые слова, которые не начинаются с восклицательного знака, являются определенными в схеме. Ключевые слова, которые начинаются с восклицательного знака, являются определенными пользователем. Определенное пользователем ключевое слово является идентификатором для поименованного типа (типа данных объекта или определенного типа) в EXPRESS-схеме, управляющей структурой обмена. Смысл ключевого слова, определенного пользователем, является предметом соглашения между партнерами, использующими структуру обмена.

6.4 Кодирование простых типов данных

В структурах обмена используют кодирование шести простых типов данных: целое (integer), вещественное (real), строка (string), имя экземпляра объекта (entity instance name), перечисление (enumeration) и двоичное (binary)*.

6.4.1 Целое (Integer)

Целое должно быть закодировано как последовательность из одной или нескольких цифр, согласно таблице 2, которой может (но необязательно) предшествовать *знак плюс* "+" или *минус* "–". Целые числа должны быть выражены в десятичной системе счисления. Если с целым числом ни один знак не связан, то целое число считается положительным.

Примеры

Верное представление целого в файле	Значение
16	Положительное 16.
+12	Положительное 12.
–349	Отрицательное 349.
012	Положительное 12.
00	Ноль.

Неверное представление целого в файле	Ошибка
26 54	Содержит *пробелы.*
32.0	Содержит *точку.*
+ 12	Содержит *пробел* между *знаком плюс* и цифрами.

6.4.2 Вещественное (Real)

Вещественное должно быть закодировано, как указано в таблице 2. Код должен состоять из десятичной мантиссы, за которой (необязательно) следует десятичный показатель степени. Десятичная мантисса состоит в порядке следования из необязательного *знака плюс* "+" или *минус* "–", последовательности из одной или более цифр, *точки* последовательности из нуля или нескольких цифр. Десятичный показатель степени состоит из *прописной буквы E,* за которой следует необязательный *знак плюс* "+" или *минус* "–" с одной или несколькими цифрами.

Примечания

1 В настоящем стандарте не сделано никаких попыток выразить концепцию точности. Когда необходимо указать значение точности, посылающая и принимающая стороны должны достигнуть соглашения по этому вопросу. Там, где точность требуется как элемент описания типа данных объекта, ее значение должно быть включено в определение типа данных объекта в EXPRESS-схеме.

2 При определенных условиях было замечено, что передача файлов открытым текстом по электронной почте повреждает точку в значении вещественного числа. Рекомендации см. в А.2.2.

Примеры

Верное представление вещественного	Значение
+0.0E0	0.0
−O.OE-O	0.0
1.5	1.5
–32.178E+02	–3217.8
0.25E8	25 миллионов
0.E25	0.
2.	2.
5.0	5.0

Неверное представление вещественного	Ошибка
1.2E3.	В обозначении показателя степени не допускается десятичная точка.
1E05	В обозначении мантиссы требуется десятичная точка.
1,000.00	Запятая не допускается.
3.E	В обозначении показателя степени должна быть хотя бы одна цифра.
.5	Десятичной точке должна предшествовать хотя бы одна цифра.
1	В обозначении мантиссы требуется десятичная точка.

6.4.3 Строка (String)

6.4.3.1 Структура строки

Строка должна быть закодирована как *апостроф* " ' ", за которым следует ноль или несколько символов из основного алфавита, и заканчиваться апострофом " ' ". Нулевая строка (строка нулевой длины) должна быть закодирована последовательностью из двух *апострофов* " '' ". Внутри строки единичный апостроф должен быть закодирован как два последовательных *апострофа*. Внутри строки единичная *косая обратная черта* "\" должна быть закодирована как *две косые обратные черты* "\\".

Как указано в 5.2, октетное представление символов в кодовых точках от U + 0080 до U + 10FFFF задается UTF-8. Когда требуется совместимость с предыдущими редакциями настоящего стандарта, эти символы могут быть закодированы как шестнадцатеричные цифры (см. HEX в таблице 2) с использованием управляющих директив, определенных в 6.4.3.3.

Дополнительные символы должны быть закодированы с использованием шестнадцатеричных цифр, как определено в 6.4.3.2, 6.4.3.3 и 6.4.3.4. WSN управляющих директив для закодированных строк приведена в таблице 4.

Примечание — При определенных условиях было обнаружено, что передача четких текстовых файлов по электронной почте повреждает *точку* в строковом значении. Рекомендации см. в А.2.2.

Таблица 4 — Управляющие директивы для строк

```
CONTROL_DIRECTIVE = PAGE | ALPHABET | EXTENDED2
                  | EXTENDED4 | ARBITRARY .

PAGE = REVERSE_SOLIDUS "S" REVERSE_SOLIDUS LATIN_CODEPOINT .

ALPHABET = REVERSE_SOLIDUS "P" UPPER REVERSE_SOLIDUS .

EXTENDED2 = REVERSE_SOLIDUS "X2" REVERSE_SOLIDUS
            HEX_TWO { HEX_TWO } END_EXTENDED .

EXTENDED4 = REVERSE_SOLIDUS "X4" REVERSE_SOLIDUS
            HEX_FOUR { HEX_FOUR } END_EXTENDED .

END_EXTENDED = REVERSE_SOLIDUS "X0" REVERSE_SOLIDUS .

ARBITRARY = REVERSE_SOLIDUS "X" REVERSE_SOLIDUS HEX_ONE .

HEX_ONE = HEX HEX .

HEX_TWO = HEX_ONE HEX_ONE .

HEX_FOUR = HEX_TWO HEX_TWO .
```

6.4.3.2 Кодирование полного алфавита по стандартам серии ИСО/МЭК 8859 внутри строки

В стандартах серии ИСО/МЭК 8859 G (x/y) является обозначением символа в "колонке" x "строке"y, т. е. значением кода (16 · x) + y в таблице кодов. Каждая часть ИСО/МЭК 8859 (ИСО/МЭК 8859-1 — ИСО/МЭК 8859-9) идентична кодовым точкам ИСО/МЭК 10646 от U+0000 до U+007F в положениях от G (02/00) до G (07/14). Стандарты серии ИСО/МЭК 8859 отличаются символами расширенного набора символов — позициями от G (10/00) до G (15/14). Для того чтобы включить в строку символы из расширенного набора, необходимо использовать директивы управления.

Примечание — Директивы управления, описанные в данном разделе, сохраняются для совместимости с предыдущими изданиями настоящего стандарта. Рекомендуется преобразовать все символы ИСО/МЭК 8859 в соответствующие значения ИСО/МЭК 10646.

Директиву управления PAGE — *обратная косая черта, прописная буква S, обратная косая черта* ("\S\"), за которой следует символ LATIN_CODEPOINT (см. таблицу 1) — используют в строке для того, чтобы позволить символу основного алфавита представить символ в соответствующей позиции расширенного алфавита ИСО/МЭК 8859. Директиву управления PAGE следует интерпретировать в

строке как одиночный символ G [(x+8)/y], где G(x/y) — символ основного алфавита, следующего за "\S\". Таким образом, если символ основного алфавита имеет значение кода v, то его следует интерпретировать как символ со значением кода v 4 + 128.

Для того чтобы указать, что только в данной строке последующие директивы управления *обратная косая черта, прописная буква S, обратная косая черта* будут интерпретироваться как ссылки на расширенный алфавит, определенный в том стандарте серии ИСО 8859, который определяется значением UPPER, должна быть использована директива управления *обратная косая черта, прописная буква P*, UPPER, *обратная косая черта*. *Прописная буква* (обозначенная как UPPER) должна быть одной из следующих: "A", "B", "C", "D", "E", "F", "G", "H", "I". В данном контексте *прописная буква A* определяет ИСО/МЭК 8859-1; *прописная буква B* — ИСО/МЭК 8859-2 и т. д. Если данная директива управления не появляется в строке, подразумевается значение "A", т. е. должен быть расширенный алфавит, который определен в ИСО/МЭК 8859-1.

Примеры

Хранящаяся строка	Содержание	Комментарии
'CAT'	CAT	
'Don''t'	Don't	
''''	'	Апостроф.
''	string of length zero	Строка нулевой длины.
'\S\Drger'	Ärger	
'h\S\ttel'	hôtel	
'\PE\\S*\S\U\S\b'	Њет	Кириллица. 'Нет'.

6.4.3.3 Кодирование внутри строки набора символов из стандартов серии ИСО/МЭК 10646

В настоящем стандарте указаны директивы управления, позволяющие кодировать символы ИСО/МЭК 10646 как последовательность шестнадцатеричных символов. Эти директивы управления могут использоваться вместо UTF-8 кодированных символов, когда требуется совместимость с предыдущими редакциями кодирования структуры обмена.

Директива управления *обратная косая черта, латинская прописная буква X, цифра два, обратная косая черта* "\X2\" указывает, что далее следует очередная последовательность из кратных четырем шестнадцатеричных символов. Каждая последовательность из кратных четырем шестнадцатеричных символов должна интерпретироваться как 16-битное число, дающее целочисленное положение в кодовом пространстве UCS.

Директива управления *обратная косая черта, латинская прописная буква X, цифра четыре, обратная косая черта* "\X4\" указывает, что далее следует очередная последовательность из кратных восьми шестнадцатеричных символов. Каждая последовательность из кратных восьми шестнадцатеричных символов должна интерпретироваться как 32-битное число, дающее целочисленное положение в кодовом пространстве UCS.

Директива управления *обратная косая черта, латинская прописная буква X, цифра ноль, обратная косая черта* "\X0\" должна использоваться для указания конца шестнадцатеричной последовательности символов "\X2\" или "\X4\".

Примечание — Такое использование восьми шестнадцатеричных символов в кодировке "\X4\" предшествует ограничению кодового пространства UCS максимальным значением 10FFFF. Первые два символа в каждой группе из восьми символов всегда будут равны нулю.

Примеры

Хранящаяся строка	Кодовая точка		Символ
'\X2\03C0\X0\'	U+03C0		*строчная греческая буква пи* (π)
'\X2\03B103B203B3\X0\'	U+03B1 U+03B3	U+03B2	*строчные греческие буквы альфа, бета и гамма* (αβγ)
'\X4\001F638\X0\'	U+1F638		*ухмыляющаяся кошачья морда с улыбающимися глазами (*символ эмоции, 😸*)*
'\X4\001F638001F596\X0\'	U+1F638 U+1F596		*ухмыляющаяся кошачья морда с улыбающимися глазами, поднятая рука с промежутком между средним и безымянным пальцами (два символа эмоции,* 😸 🖖*)*

6.4.3.4 Кодирование от U + 0000 до U + 00FF в строке

Директива управления *обратная косая черта, латинская прописная буква X обратная косая черта* "\X\", за которой следуют два шестнадцатеричных символа, должна кодировать точку кода UCS в диапазоне от U + 0000 до U + 00FF. Два шестнадцатеричных символа должны интерпретироваться как 8-битное число, дающее целочисленное положение в кодовом пространстве UCS.

Данная директива управления должна использоваться для кодовых точек UCS от U + 0000 до U + 001F и кодовой точки U + 007F. Эта директива управления может использоваться вместо UTF-8 кодированных кодовых точек от U + 0080 до U + 00FF, когда требуется совместимость с более ранними редакциями кодирования структуры обмена.

Примечание — Символы, определенные в ИСО/МЭК 10646 и ИСО/МЭК 8859-1, идентичны в этом диапазоне.

Примеры

Хранящаяся строка	Содержание	Комментарии
'see \X\A7 4.1'	См. § 4.1	Содержит знак параграфа.
'line one\X\0Aline two'	строка один строка два	Содержит управляющий символ *перевода строки.*

6.4.3.5 Максимальная длина строки

Максимальная длина строки, сохраняемая в структуре обмена, ограничена 32769 8-битными байтами, включая начальный и конечный *апострофы.* Если в хранящуюся строку включены *кавычки, обратная косая черта, апострофы,* директивы управления печатью (см. раздел 12) или символы, закодированные в соответствии с 6.4.3.2, 6.4.3.3 или 6.4.3.4, максимальная длина действительного содержания строки будет меньше, чем 32767 графических символов. Действительным содержанием является последовательность графических символов, полученная после того, как будут выполнены соглашения по кодированию.

6.4.4 Имена вхождений (Occurrence names)

Имя вхождения должно быть именем экземпляра константы, именем значения константы, именем экземпляра объекта или именем экземпляра значения.

Примечание — Данная редакция настоящего стандарта позволяет именовать постоянные значения, постоянные объекты, экземпляры значений и экземпляры объектов и ссылаться на них в структуре обмена. Предыдущие редакции разрешали именовать и ссылаться только на экземпляры сущности (см. 4.3).

6.4.4.1 Имена экземпляра константы

Имя экземпляра константы должно быть закодировано как *знак номера* "#", за которым следует символ UPPER, за которым следует последовательность символов UPPER или DIGIT.

Имена экземпляров констант — это ссылки на экземпляры сущностей, определенные в EXPRESS-схеме. Если в схеме **file_schema** структуры обмена определено несколько схем EXPRESS, то имя экземпляра константы должно ссылаться на экземпляр объекта, определенного в первой схеме (см. 8.2.4).

WSN для имен экземпляров констант приведена в таблице 2 в правиле подстановки CONSTANT_INSTANCE_NAME.

Примеры

Допустимые выражения имен	Значение
#FARADAY	Ссылка на константу FARADAY (Фарад), определение которой содержится в EXPRESS-схеме.
#INCH	Ссылка на константу INCH (дюйм), определение которой содержится в EXPRESS-схеме.
Недопустимые выражения имен	**Ошибка**
#23	Наименование (идентификатор) начинается с цифры.
#INCHES	INCH определяется как экземпляр ENTITY в EXPRESS-схеме.
# Pie	Все буквы должны быть приведены в верхнем регистре.

Имена экземпляров констант могут быть использованы только в правилах RHS_OCCURRENCE (см. таблицу 2).

6.4.4.2 Имена значений константы

Имя значения константы должно быть закодировано как *знак* "@", за которым следует символ UPPER, за которым следует последовательность символов UPPER или DIGIT.

Имена значений констант — это ссылки на значения сущностей, определенных в EXPRESS-схеме. Если в схеме **file_schema** структуры обмена определено несколько схем EXPRESS, то имя значения константы должно ссылаться на значение, определенное в первой схеме (см. 8.2.4).

WSN для имен значений констант приведен в таблице 2 в правиле подстановки CONSTANT_VALUE_NAME.

Примеры

Допустимые выражения имен	Значение
@PI	Ссылка на значение константы PI, определение которой содержится в EXPRESS-схеме.
@E	Ссылка на значение константы E, определение которой содержится в EXPRESS-схеме.
Недопустимые выражения имен	**Ошибка**
@23	Наименование начинается с цифры.
@INCH	INCH (дюйм) — это наименование объекта, определение которого содержится в EXPRESS-схеме.
@Pie	Все буквы должны быть приведены в верхнем регистре.

Имена значений констант могут быть использованы только в правилах RHS_OCCURRENCE (см. таблицу 2).

6.4.4.3 Имена экземпляров объекта

Имя экземпляра объекта должно быть закодировано как *знак номера* "#", за которым следует целое без знака. Целое должно представлять любую комбинацию из одной или нескольких десятичных цифр. По меньшей мере одна цифра не должна быть "0". Предшествующие нули в имени экземпляра объекта не имеют значения. Имя экземпляра объекта не должно использовать то же целое число, что и имя экземпляра значения.

Примечания

1 Целочисленные значения для ENTITY_INSTANCE_NAME и VALUE_INSTANCE_NAME не могут перекрываться, поскольку на оба типа можно ссылаться с помощью URI, например "<abc.stp # 123>" (см. 10.2.7).

2 Начальные нули в именах экземпляров объектов игнорируются, поэтому "# 001" является тем же идентификатором, что и "# 1".

WSN для имен значений констант приведен в таблице 2 в правиле подстановки ENTITY_INSTANCE_NAME.

Примеры

Правильное выражение имени	Значение
#12	Именует экземпляр объекта или ссылается на экземпляр объекта с идентификатором 12.
#023	Именует экземпляр объекта или ссылается на экземпляр объекта с идентификатором 23.
Неправильное выражение имени	**Ошибка**
#Faraday	Содержит не числовой символ.
#439A6	Содержит не числовой символ.
#+23	Содержит знак '+'.

#00.1	Содержит десятичную точку.
74	Не начинается со *знака номера.*

Имена экземпляров объектов используются в качестве ссылок на экземпляры объектов. Допустимы как прямые, так и обратные ссылки. Имя экземпляра объекта может быть определено в ссылочной секции (см. раздел 10) или секции данных (раздел 11). Имена экземпляров объектов могут использоваться в правилах LHS_OCCURRENCE и RHS_OCCURRENCE (см. таблицу 2).

6.4.4.4 Имена значений экземпляров

Имена значений экземпляров должны быть закодированы как *знак* "@", за которым следует последовательность символов DIGIT. По крайней мере один символ не должен быть "0". Стоящие впереди нули не являются значимыми. Имя значения экземпляра не должно использовать то же целое число, что и имя экземпляра сущности.

Примечание — Данная редакция настоящего стандарта позволяет назначать имена значениям экземпляров, чтобы значения могли быть определены во внешних файлах. Примеры приведены в приложении К.

WSN для имен значений экземпляров приведен в таблице 2 в правиле подстановки VALUE_INSTANCE_NAME

Примеры

Правильное выражение имени	Значение
@12	Наименование или ссылка на значение с обозначением 12.
@023	Наименование или ссылка на значение с обозначением 23.

Имена значений экземпляров используются в качестве ссылок на экземпляры объектов. Имя значения экземпляра может быть определено в ссылочной секции (см. раздел 10). Имена значений экземпляров могут использоваться в правилах LHS_OCCURRENCE и RHS_OCCURRENCE (см. таблицу 2). Имя значения экземпляра должно быть определено только в ссылочной секции.

6.4.5 Перечисляемые значения

Перечисляемое значение должно быть закодировано как последовательность *латинских прописных букв* или *цифр,* начинающаяся с *латинской прописной буквы,* ограниченная *точками.* Смысл заданного перечисляемого значения задается EXPRESS-схемой и соответствующими определениями из объявлений перечисляемого типа.

Примечание — При определенных условиях было замечено, что передача файлов открытым текстом по электронной почте повреждает *точку* в начале или конце перечисляемого значения. Рекомендации см. в А.2.2.

Примеры

Правильное перечисляемое выражение	Значение
.STEEL.	Показывает значение STEEL.
Неправильное перечисляемое выражение	**Ошибка**
.RED	Пропущена точка в конце.
.123.	Начинается не с алфавитно-цифрового символа.

6.4.6 Двоичное число (Binary)

Двоичным числом является последовательность битов (0 или I). Двоичное число должно быть закодировано так, как определено следующей процедурой:

- подсчитывают количество битов в последовательности. Называют результат *р;*
- определяют число *п*, $0 \le п \le 3$, так что $к = р + п$ является кратным четырем;
- дополняют слева двоичное число *п* нулевыми битами. Разделяют последовательность на группы по четыре бита;
- перед последовательностью вводят четырехбитное представление *п;*

- если десятичный эквивалент 4-битной группы 9 или меньше, чтобы получить 8-битный байт, добавляют к десятичному значению 48; если десятичный эквивалент 4-битной группы больше 9, чтобы получить 8-битный байт, добавляют к десятичному значению 55.

Примечание — Это преобразование двоичного числа в шестнадцатеричное;

- закодированное двоичное число состоит из *k*/4+1 шестнадцатеричных цифр. Первая цифра является значением *n.* За ней следуют шестнадцатеричные цифры, представляющие двоичное число;
- двоичное число отделяют *кавычками* " " ".

Примеры

Двоичное значение	Представление
'null' or 'empty'	"0"
0	"30"
1	"31"
111011	"23B"
100100101010	"092A"

6.5 Кодирование секций привязки, ссылочной секции и подписи

В секциях привязки, ссылочной секции и подписи используются следующие кодировки.

6.5.1 Ресурс

Ресурс кодируется как URI, которому предшествует *знак меньше*, "<" и за которым следует *знак больше*, ">".

WSN для ресурсов приведен в таблице 2 в правиле RESOURCE.

Примечание — В разделе привязки ресурс находится справа от *знака равенства* ("="), а имя привязки — слева, см. 6.5.4.

Примеры

Правильное выражение в секции привязки	Значение
<picture> = <a.jpeg>;	Устанавливает для ресурса <a.jpeg> имя привязки "picture" (изображение).
<BOM> = <b.xml#123>;	Устанавливает для ресурса <b.xml#123> имя привязки "BOM" (конструкторская спецификация).

Примечание — Ресурс в ссылочной секции должен реализовываться в виде экземпляра сущности или экземпляра значения. Процесс реализации см. в разделе 10.

Примеры

Правильное выражение в ссылочной секции	Значение
#10 = <a#b>;	Экземпляру объекта 10 присваивается значение объекта, обозначенного ресурсом <a#b>.
@20 = <c#d>;	Экземпляру объекта 20 присваивается значение объекта, обозначенного ресурсом <c#d>.

6.5.2 Универсальный идентификатор ресурса (URI)

Лексема UNIVERSAL_RESOURCE_IDENTIFIER таблицы 2 должна соответствовать требованиям, определенным в IETF (см. 3.1.7.1).

Примеры

Внешняя ссылка	Пример использования
<gost.e-ecolog.ru >	Ссылка на хранящееся в файле STEP-NC, расположенном по заданному интернет-адресу, описание заготовки.
<building.ifc#first_floor>	Ссылка на хранящееся на настоящем сервере в формате IFC описание пола в здании.
<file:///c:/users/jt_files/assembly.jt.#first_shape>	Ссылка на описание формы, которое содержится в файле формата JT.

6.5.3 Идентификатор фрагмента URI

Лексема URI_FRAGMENT_IDENTIFIER таблицы 2 — это имя, следующее за *знаком номера* "#" в универсальном идентификаторе ресурса.

Примеры

Универсальный идентификатор ресурса	Фрагмент идентификатора	Пример использования
<gost.e-ecolog.ru >	tool_tip	Обозначение фрагмента для точки на наконечнике режущего инструмента.
<#first_floor>	first_floor	Обозначение фрагмента для этажа в текущей обменной структуре.
<gost.e-ecolog.ru gost.e-ecolog.ru >	11D3-9A0C-0305E82C3301	ние фрагмента (см. приложение G).

6.5.4 Имя привязки

Имя привязки должно быть закодировано как идентификатор фрагмента URI, которому предшествует *знак меньше*, "<" и за которым следует *знак больше*, ">". По крайней мере один символ в идентификаторе фрагмента URI, который ссылается на имя привязки, не должен быть цифрой.

Примечание — Идентификаторы фрагментов URI, определенные как цифры, считаются ссылками на имена экземпляров в структурах обмена, определенных в предыдущих редакциях BCJ 10303-21. См. 10.2.7.

Имя привязки, соответствующее требованиям приложения G, является универсальным уникальным идентификатором (UUID).

Примечание — Привязки, определенные UUID, могут быть найдены без URI, поскольку они универсально уникальны. См. 10.2.2.

WSN для имен привязки приведена в таблице 2 в правиле ANCHOR_NAME. Имена привязки используются для определения идентификаторов, на которые можно ссылаться извне (см. раздел 9).

Примеры

Правильное выражение в секции привязки	Значение
<a> = 3.142;	Значению 3.142 присваивается имя привязки "a".
<b> = @10;	Значению @10 присваивается имя привязки "b".
<c> = #20;	Экземпляру объекта #20 присваивается имя привязки "c".
<ad3f1724-19cf-4d19-94ef-eed90b7b4dde> = 2.71828;	Значению 3.142 присваивается имя привязки с UUID "ad3f1724-19cf-4d19-94ef-eed90b7b4dde".
<2f0cb220-355d-11e5-a2cb-0800200c9a66> = @30;	Значению @30 присваивается имя привязки с UUID "2f0cb220-355d-11e5-a2cb-0800200c9a66".
<3f553e90-355d-11e5-a2cb-0800200c9a66> = #40;	Экземпляру объекта #40 присваивается имя привязки UUID "3f553e90-355d-11e5-a2cb-0800200c9a66".

6.5.5 Имя тега

Имя тэга должно быть закодировано как последовательность символов UPPER, LOWER и DIGIT. Первый символ должен быть символом UPPER или LOWER.

WSN для тэга имени приведена в таблице 2 в правиле TAG_NAME. Имена тэгов связывают дополнительную информацию с привязкой. Эта информация не является частью информационной модели (см. 9.2.8).

Примечание — В данном издании настоящего стандарта разрешены имена тэгов, чтобы программисты могли создавать структуры данных для оптимизации обходов, когда информационная модель распределяется по множеству структур обмена, связанных привязками и ссылками.

Пример

Правильное выражение в секции привязки	Значение
<plate_edge> = #20 {preparation:<WELD_DC.XML>}	Представляющий ребро объект #20 связывается с файлом WELD_DC.XML, используя имя тэга "preparation" (подготовка).

6.5.6 Base64

Лексема BASE64 таблицы 2 представляет собой данные, кодированные в соответствии с требованиями IETF (3.1.7.5). Base64 используется для кодирования подписей и дайджестов сообщений.

Пример

Кодирование дайджестов сообщений с использованием Base64

873b48e9dd16ec9c7a8423faba7e75a7a9d19ea07abce2808d94b3176ee8bd60

7 Структурированные типы данных

В структуре обмена допустимы два типа списка данных.

7.1 Параметр LIST (список)

Как определено в таблице 3, LIST является последовательностью (возможно пустой) из PARAMETER (параметров), каждый из которых может быть:

- кодированием простого типа, как описано в 6.3, или
- специальной лексемой *знак доллара* "$", или
- TYPED_PARAMETER, представляющим экземпляр выбранного типа (см. 12.1.8), или
- LIST, представляющим экземпляр (вложенного) структурированного типа.

Данный список может содержать более одной из вышеперечисленных форм. В структуре обмена список начинается с *левой круглой скобки* "(" и заканчивается *правой круглой скобкой* ")". Экземпляры разделяют *запятыми.* Список может быть вложенным на любую глубину.

Пример

Структурированный тип данных	Представление
Список целых	(0,1,2,3,7,2,4)
Список строк	('CAT', 'HELLO')
Список списков вещественных значений	((0.0, 1.0, 2.0), (3.0, 4.0, 5.0))
Список списков вещественных значений	((0.0, 1.0, 2.0), ( ))

В последнем списке списков вещественных значений второй вложенный список пустой.

7.2 Список элементов привязки

Как определено в таблице 3, ANCHOR_ITEM_LIST может быть определен в секции привязки. Список должен содержать последовательность значений, каждое из которых может быть:

- кодированием простого типа, как описано в 6.3, или
- кодированием ресурса, как описано в 6.5.1, или
- специальной лексемой *знак доллара* "$", или
- ANCHOR_ITEM_LIST, представляющим экземпляр (вложенного) структурированного типа.

Данный ANCHOR_ITEM_LIST может содержать более одной из вышеуказанных форм. В структуре обмена ANCHOR_ITEM_LIST начинается с *левой круглой скобки* "(" и заканчивается соответствующей *правой круглой скобкой* ")". Экземпляры разделяются запятыми. Элементы ANCHOR_ITEM_LIST могут быть вложены на любую глубину.

Пример

Структурированный тип данных	Представление
Список URI	(<abc#d>, <def.xml>)
Список целых	(1, 2, 3)

8 Заголовочная секция

8.1 Структура заголовочной секции

Заголовочная секция содержит информацию, которая относится ко всей структуре обмена. Эта секция должна быть представлена в каждой структуре обмена.

Секция должна начинаться со специальной лексемы "HEADER:" и заканчиваться специальной лексемой "ENDSEC;".

Заголовочная секция должна содержать один экземпляр каждого из следующих объектов: **file_description**, **file_name** и **file_schema**, и они должны располагаться в этой последовательности. Экземпляры объектов **schema_population**, **file_population**, **section_language** и **section_context** могут появиться после объекта **file_schema**.

Примечание — В приложениях H и I представлены примеры заголовочной секции.

Если имеются экземпляры определенных пользователем объектов заголовочной секции, то они должны появляться после обязательных экземпляров объектов заголовочной секции. Синтаксис экземпляров объектов заголовочной секции приведен в WSN в таблице 3. Каждое имя объекта должно быть преобразовано в KEYWORD (ключевое слово) правила HEADER_ENTITY. Раздел 12 определяет преобразование простых и составных (агрегатных) типов данных в PARAMETER_LIST (список параметров) для значений атрибутов данных экземпляров объектов.

8.2 Декларации заголовочной секции

8.2.1 Схема заголовочной секции

Данный подраздел определяет объекты и типы заголовочной секции, которые появляются в заголовочной секции структуры обмена. Объекты заголовочной секции определены с помощью языка EXPRESS.

EXPRESS-спецификация:

```
*)
SCHEMA header_section_schema;
TYPE exchange_structure_identifier = STRING;
END_TYPE
*)
```

Данная схема определяет объекты заголовочной секции, которые специфичны для процесса передачи данных об изделии с использованием структуры обмена.

Примечание — Тип данных **exchange_structure_identifier** соответствует типу идентификатора, установленному в ИСО 10303-41, но должен быть задан отдельно, чтобы обеспечить независимость требований настоящего стандарта от моделей данных, описанных в группе интегрированных ресурсов стандартов серии ИСО 10303.

8.2.2 Объект file_description

Объект **file_description** определяет версию настоящего стандарта, использованную для создания структуры обмена, а также ее содержание.

EXPRESS-спецификация:

```
*)
ENTITY file_description;
description : LIST [I : ?] OF STRING (256);
implementation_level : STRING (256);
END.ENTITY;
(*
```

Описание атрибутов:

description — неформальное описание содержимого структуры обмена;

implementation_level — обозначение требований, которым соответствует кодирование в данной структуре обмена, и любых соответствующих вариантов, применяемых при кодировании. Значение этого атрибута должно указывать соответствие данной версии настоящего стандарта наличием значения "4;1" или "4;2", или "4;3" (см. 4.3). Для структур обмена, принадлежащих к классу соответствия 1, зна-

чение должно быть задано в виде "4;1". Для структур обмена, принадлежащих к классу соответствия 2, значение должно быть "4;2". Для структур обмена, принадлежащих к классу соответствия 3, значение должно быть "4;3".

Если встречаются нижеперечисленные ограничения, то для указания соответствия настоящему стандарту может быть использовано значение "3;1":

- структура обмена не должна содержать секцию привязки;
- структура обмена не должна содержать ссылочную секцию;
- заголовочная секция структуры обмена не должна содержать объекта SCHEMA_POPULATION;
- строковые лексемы в структуре обмена должны кодировать только символы с U + 0080 по U + + 10FFFF по директивам управления "\X2\" и "\X4\", определенным в 6.4.3.3;
- структура обмена не должна содержать секцию подписи;
- структура обмена не должна ссылаться на сокращенные наименования языка EXPRESS.

При использовании значение "3;1" должно обозначать структуры обмена, относящиеся к классу синтаксического соответствия 1 издания 2.

Если встречаются нижеперечисленные ограничения, тогда для указания соответствия настоящему стандарту может быть использовано значение "2;1":

- конкретная структура обмена должна содержать единственную секцию данных, а ключевое слово "DATA" не должно располагаться за PARAMETER_LIST;
- заголовочная секция структуры обмена не должна содержать объектов FILE_POPULATION;
- заголовочная секция структуры обмена не должна содержать объектов SECTION_LANGUAGE;
- заголовочная секция структуры обмена не должна содержать объектов SECTION_CONTEXT;
- перечисляемые значения (ENUMERATION) на языке EXPRESS не должны кодироваться с использованием сокращенных наименований.

При использовании значение "2;1" должно обозначать структуры обмена, относящиеся к классу синтаксического соответствия 1 издания 1.

Примечания

1 Общая форма для значения имеет вид "v;cc", где v — номер версии настоящего стандарта, как указано в приложении C, а cc — код класса соответствия. В будущих версиях настоящего стандарта могут быть указаны дополнительные значения v и cc.

2 Использование "3;1" предусмотрено для обеспечения совместимости с реализациями, основанными на предыдущей версии стандарта ИСО 10303-21:2002, а использование "2;1" предусмотрено для поддержки совместимости с реализациями, основанными на предыдущей версии стандарта ИСО 0303-21:1994.

3 Значения "2;2" и "3;2" использовались более ранними редакциями для указания кодировки, которая использовала внешнее сопоставление для всех экземпляров сущностей.

8.2.3 Объект file_name

Объект **file_name** представляет доступную для прочтения человеком информацию о структуре обмена. Содержание атрибутов данного объекта, за исключением атрибута **time_stamp**, не определяется настоящим стандартом.

EXPRESS-спецификация:

```
*)
ENTITY file_name;
  name                 : STRING (256) ;
  time_stamp           : time_stamp_text ;
  author               : LIST [ 1 : ? ] OF STRING (256) ;
  organization         : LIST [ 1 : ? ] OF STRING (256) ;
  preprocessor_version : STRING (256) ;
  originating_system   : STRING (256) ;
  authorization        : STRING (256) ;
END_ENTITY;

TYPE time_stamp_text = STRING(256);
END_TYPE;
(*
```

Описание атрибутов:

name — последовательность графических символов, используемых для именования данной структуры обмена. Две структуры обмена с одинаковым именем должны использовать совместимые представления для привязки с одинаковым именем привязки.

Использование в имени идентификатора объекта рекомендуется, но не требуется, поскольку оно обеспечивает однозначную идентификацию структуры привязки.

Если используется идентификатор объекта, то он должен иметь форму, указанную в ИСО/МЭК 8824-1. Использование идентификаторов объектов в рамках настоящего стандарта описано в разделе 3 ИСО 10303-1.

Примеры

Атрибут **name**	Пример использования
brep_shape	В обменной структуре имеются точки привязки для граничной модели формы.
PMI{ 1 2 10303 501 0 1 1 1 }	В обменной структуре имеются точки привязки для версии информации об изготовлении изделия, определяемой идентификатором объекта (OID);

time_stamp — дата и время, показывающие, когда была создана структура обмена. Содержание строки должно соответствовать расширенному формату полной календарной даты, как определено в ИСО 8601, объединенному с расширенным форматом времени дня, установленным в 4.3.1.1 или 4.3.3 ИСО 8601. Дата и время должны быть разделены *прописной буквой Т*, как определено в 4.4.1 ИСО 8601. Другие форматы по 4.3.1.1 и 4.3.3 позволяют дополнительно включать определитель часового пояса;

author — имя и почтовый адрес лица, ответственного за создание структуры обмена;

organization — группа или организация, с которой связан автор;

preprocessor_version — система, используемая для создания структуры обмена, включая имя и версию системного изделия;

originating_system — система, от которой выданы данные в эту структуру обмена;

authorization — имя и почтовый адрес лица, уполномоченного посылать структуру обмена.

Пример

Элемент **Time_stamp**	Полный расширенный формат
Календарная дата 12 апреля 1993	1993-04-12
Время суток 15 часов 27 минут 46 секунд	15:27:46
Часовой пояс 5 часов на запад от Гринвича	Поле часового пояса необязательно: -05:00
Вышеупомянутые дата и время, закодированные в поле штампа времени	1993-04-12T15:27:46-05:00

8.2.4 Объект file_schema

Объект **file_schema** указывает EXPRESS-схемы, в которых определены экземпляры объектов в секции данных. Атрибут **schema_identifiers** должен состоять из списка строк, каждая из которых должна содержать имя схемы, за которым следует необязательный идентификатор объекта, присвоенный этой схеме.

Если имя схемы содержит *строчные буквы*, то они должны быть преобразованы в соответствующие *прописные буквы*. В строках **schema_name** должны быть только *прописные буквы*.

Если известен идентификатор объекта, он должен иметь форму, установленную в ИСО/МЭК 8824-1. Использование идентификаторов объектов в стандартах серии ИСО 10303 описано в разделе 3 ИСО 10303-1. Рекомендуется, по возможности, применять идентификатор объекта, так как это обеспечивает однозначное обозначение схемы.

Примечание — Общим форматом представления идентификатора описываемого предмета (физического объекта) являются отдельные целые числа. Последовательность этих чисел представляют в соответствующих *фигурных скобках* ("{", "}").

EXPRESS-спецификация ^

```
*)
ENTITY file_schema;
  schema_identifiers : LIST [1:?] OF UNIQUE schema_name;
END_ENTITY;

TYPE schema_name = STRING(1024);
END_TYPE;
(*
```

Описание атрибутов:

schema_identifiers — схемы, которые определяют экземпляры объектов в секции данных.

Пример — Нижеописанный экземпляр EXPRESS-схемы назван 'CONFIG_CONTROL_DESIGN':

FILE_SCHEMA (('CONFIG_CONTROL_DESIGN'));

Следующий экземпляр использует идентификатор описываемого предмета (физического объекта) 'AUTOMOTIVE_DESIGN' для указания конкретной версии EXPRESS-схемы:

FILE_SCHEMA (('AUTOMOTIVE-DESIGN { 1 0 10303 214 1 1 1 }'));

8.2.5 Объект schema_population

Объект **schema_population** определяет совокупность экземпляров, составляющих совокупность схем структуры обмена, с целью определения соответствия схемы первой схеме **file_schema** (см. 8.2.4). Заголовочная секция структуры обмена должна содержать не более одного экземпляра **schema_population**.

Данная коллекция схемы должна содержать все экземпляры объекта во всех секциях данных в структуре обмена. Дополнительные члены коллекции определяются следующим образом:

- если присутствует объект **schema_population**, то коллекция схемы включает в набор **external_file_locations** совокупность схем каждой структуры обмена;
- если структура обмена включает ссылочную секцию, то набор схем должен включать в себя набор схем всех структур обмена, на которые ссылаются URI в ссылочной секции;
- каждый экземпляр объекта должен отображаться в наборе схем не более одного раза.

Примечание — Определение заполнения схемы является транзитивным. Если файл А ссылается на файл В, а файл В ссылается на файл С, то совокупность схем А включает в себя совокупности схем как В, так и С.

Пример — Файл может иметь несколько ссылок на один и тот же экземпляр сущности из-за повторяющихся ссылок, универсальных идентификаторов ресурсов, которые обращаются к одному и тому же адресу, или по другим причинам. В таких случаях экземпляр появляется только один раз в наборе схем.

Каждый объект **external_file_location** должен быть представлен тремя строками, которые определяют адрес, представленную объектом **time_stamp** необязательную метку времени и необязательную копию представленного объектом **message_digest** дайджеста сообщений (см. раздел 14).

Если файл по адресу не включается в структуру обмена, то экземпляры из этого файла не должны быть включены в заполнение схемы.

Если метка времени включена и ее значение описывает дату и время после даты и времени в **file_name** заголовка ссылочной структуры обмена, то ссылочная структура обмена не должна быть изменена с момента создания этой ссылки.

Если включен объект **message_digest**, то он проверяет целостность файла ссылки. Дайджест сообщения выдается только в том случае, если структура обмена имеет хотя бы одну секцию подписи. Дайджест сообщения должен использовать алгоритм хеширования, выбранный для первой секции подписи.

EXPRESS-спецификация:

```
*)
ENTITY schema_population;
```

```
  external_file_identifications : LIST [ 1 : ? ] OF LIST [1:3] OF STRING;
END_ENTITY;
(*
```

Описание атрибутов:

external_file_identifications — список внешних адресов файлов, которые должны быть включены в набор схем этого файла. Каждый адрес должен быть представлен в виде тройки строк. Если вторая или третья строка не задана, то она будет представлена как нулевое значение "$".

Первая строка должна быть адресом ссылочной структуры обмена, представленной как универсальный идентификатор ресурса (см. 6.5.1), закодированный как STRING (см. 6.4.3).

Во второй строке должно быть представлено значение объекта **time_stamp**, описывающего дату и время последнего посещения внешнего ресурса. Строка с описанием даты и времени должна соответствовать представленным в 8.2.3 требованиям к **time_stamp_text**.

Третья строка должна быть представлена значением объекта **message_digest** для ссылочного файла. В качестве **message_digest** используется строка в кодировке Base64 (см. 3.1.7.5).

Примечание — Включение ссылочного файла **message_digest** в коллекцию схемы означает, что при подписании этого файла ссылочный файл также подписывается.

Пример — Создание экземпляра для заполнения схемы.

```
SCHEMA_POPULATION ((('gost.e-ecolog.ru                    , '2012-12-09T17: 00: 00', $),
('gost.e-ecolog.ru                    , '2013-01-11T19: 00: 00', '0c1c87c6e731829d36eb36b6f16deaeefb7a
f422033372c8fb5e94fb0af346b'),
('gost.e-ecolog.ru                    , '2013-01-11T19: 00: 00', 'f243b19fb3c9f4f1a71edb9ec6390d1598
7b443ade7ad3081a174ff421ef8fe7')));
```

Примечание — Файлы, не являющиеся структурами обмена, могут быть включены в коллекцию схемы, чтобы их содержимое также можно было проверить с помощью подписи.

8.2.6 Объект file_population

Объект **file_population** задает набор (коллекцию) экземпляров объектов в совокупности схем, определенной в 8.2.5, с целью определения соответствия схемы конкретной EXPRESS-схеме. Данная коллекция должна быть задана посредством алгоритма, определенного атрибутом **determination_method** для набора секций данных, указанных атрибутом **governed_sections**. Если в этом атрибуте не указано конкретное значение, данный алгоритм должен быть использован для всех секций данных в структуре обмена.

Структура обмена может содержать ноль, один или несколько экземпляров **file_population**. Имя секции данных может быть указано в атрибуте **governed_sections** нулевого, одного или нескольких экземпляров **file_population**.

Примечания

1 В Е.2 заданы три возможных метода определения данного объекта.

2 Если в атрибуте **governed_sections** не задано конкретное значение, его кодируют *знаком доллара* ("$") в соответствии с 12.2.2, а не в виде пустого списка.

EXPRESS-спецификация:

```
*)
ENTITY file_population;
  governing_schema      : schema_name;
  determination_method  : exchange_structure_identifier;
  governed_sections     : OPTIONAL SET [ 1 : ? ] OF section_name;
END_ENTITY;

TYPE section_name = exchange_structure_identifier;
END_TYPE;
(*
```

Описание атрибутов:

governing_schema — имя EXPRESS-схемы, используемой для совокупности экземпляров объектов, заданных в заголовочной секции объекта **file_population**. Данное имя может быть размещено в заголовочной секции объекта **file_schema**:

determination_method — строка графических символов, применяемая для определения алгоритма, используемого при выборе экземпляров объектов для обмена;

governed_sections — имена секций данных, использованные в качестве исходных данных для выбранного метода определения данного объекта.

8.2.7 Объект section_language

Объект **section_language** определяет по умолчанию язык описания строковых значений в секции данных. Атрибут **default_language** должен содержать наименование конкретного языка. Наименование языка должно быть закодировано в соответствии с библиографическим кодом Alpha-3, определенным в ИСО 639-2.

Атрибут **section** должен содержать наименование секции данных в структуре обмена, для которой по умолчанию задан соответствующий язык. Если в структуре обмена содержится единственная не поименованная секция данных, тогда значение атрибута **section** не определяют (см. 12.2.2). Заголовочная секция структуры обмена должна содержать по крайней мере один экземпляр объекта **section_language** с неопределенным значением атрибута **section**. При необходимости заданный в этом экземпляре по умолчанию язык должен быть использован для всех секций данных в структуре обмена, в которых не определены экземпляры других объектов **section_language**.

Пример — Примерами значений атрибута default_language являются: 'end' — для английского языка, 'fre' — для французского, 'rus' — для русского или 'ger' — для немецкого.

EXPRESS-спецификация:

```
*)
ENTITY section_language;
  section          : OPTIONAL section_name;
  default_language : language_name;
UNIQUE
  UR1: section;
END_ENTITY;

TYPE language_name = exchange_structure_identifier;
END_TYPE;
(*
```

Описание атрибутов:

section — имя секции данных, для которой применяют язык, заданный по умолчанию атрибутом **default_language**;

default_language — наименование языка, используемого для строковых значений.

8.2.8 Объект section_context

Объект **section_context** задает информацию, описывающую контексты использования экземпляров, закодированных в структуре обмена.

Атрибут **section** должен содержать имя секции данных в структуре обмена, для которой используют конкретные идентификаторы контекстов. Если в структуре обмена содержится единственная не поименованная секция данных, тогда значение атрибута **section** не определяют (см. 12.2.2). Заголовочная секция структуры обмена должна содержать по крайней мере один экземпляр объекта **section_context** с неопределенным значением атрибута **section**. При необходимости заданные в этом экземпляре идентификаторы контекстов должны быть использованы для всех секций данных в структуре обмена, в которых не определены экземпляры других объектов **section_context**.

EXPRESS-спецификация:

```
*)
ENTITY section_context;
  section             : OPTIONAL section_name;
  context_identifiers : LIST [1:?] OF context_name;
UNIQUE
  UR1: section;
END_ENTITY;

TYPE context_name = STRING;
END_TYPE;
(*
```

Описание атрибутов:

section — имя секции данных, для которой применяют контексты, заданные атрибутом **context_identifiers**:

context_identifiers — идентификаторы, содержащие информацию о контекстах экземпляров, закодированных в структуре обмена.

Примечание — В прикладном протоколе могут быть заданы символические идентификаторы для каждого класса соответствия этого протокола. Идентификаторы контекстов для секции могут задавать конкретный список классов соответствия прикладного протокола, обеспечиваемый данными из этой секции.

Примеры

1 Язык и идентификатор контекста заданы для структуры обмена в единственной непоименованной секции данных.

```
HEADER;
   .
   .
FILE_SCHEMA(('GEOMETRY'));
SECTION_LANGUAGE ($,'eng'); -----------------> A
SECTION_CONTEXT ($,('tag_a')); --------------> B
ENDSEC;
DATA;
   .
ENDSEC;
```

A — для секции данных выбран английский язык, закодированный как 'eng'.

B — для секции данных выбран признак (тег) контекста 'tag_a'.

2 Язык и идентификатор контекста заданы для структуры обмена в нескольких секциях данных.

```
HEADER;
   .
   .
FILE_SCHEMA(('GEOMETRY'));
SECTION_LANGUAGE ('DS1','ger'); -------------> A
SECTION_LANGUAGE ('DS2','epo'); -------------> B .
SECTION_LANGUAGE ($,'haw'); -----------------> C
SECTION_CONTEXT ('DS1',('tag_a','tag_b')); --> D
SECTION_CONTEXT ('DS2',('tag_c')); ----------> E
SECTION_CONTEXT ($,('tag_d')); --------------> F
ENDSEC;
DATA ('DS1',('GEOMETRY'));
   .
ENDSEC;
DATA ('DS2',('GEOMETRY'));
   .
ENDSEC;
DATA ('DS3',('GEOMETRY'));
   .
ENDSEC;
DATA ('DS4',('GEOMETRY'));
   .
ENDSEC;
```

A — для секции данных с именем 'DS1' выбран немецкий язык, закодированный как 'ger'.

B — для секции данных с именем 'DS2' выбран язык Эсперанто, закодированный как 'epo'.

C — для секций данных с именами 'DS3' и 'DS4' выбран гавайский язык, закодированный как 'haw'.

D — для секции данных с именем 'DS1' выбраны идентификаторы контекстов 'tag_a' и 'tag_b'.

E — для секции данных с именем 'DS2' выбран идентификатор контекста 'tag_c'.

F — для секций данных с именами 'DS3' и ‹DS4' выбран идентификатор контекста 'tag_d'.

```
*)
END_SCHEMA;
(*
```

8.3 Объекты заголовочной секции, определенные пользователем

В заголовочной секции могут быть помещены экземпляры объектов заголовочной секции, определенные пользователем, с конкретными ограничениями, перечисленными ниже.

a) Экземпляры объектов заголовочной секции, определенные пользователем, должны соблюдать тот же синтаксис, что и все экземпляры объектов заголовочной секции, с дополнительным требованием, что первый символ ключевого слова должен быть *восклицательным знаком* "!".

b) Атрибуты объектов заголовочной секции, определенные пользователем, должны иметь типы данных из языка EXPRESS и отображаться в заголовочной секции, как определено в разделе 12.

Пример

```
HEADER;
   .
   .
FILE_SCHEMA(('GEOMETRY'));
!A_SPECIAL_ENTITY ('ABC',123); -------> ОБЪЕКТ, ОПРЕДЕЛЕННЫЙ ПОЛЬЗОВАТЕЛЕМ
   .
   .
ENDSEC;
```

9 Секция привязки

9.1 Структура секции привязки

Секция привязки определяет внешние имена экземпляров в структуре обмена, чтобы на них можно было ссылаться.

Синтаксис секции привязки определен в таблице 3. Секция привязки является необязательной. Если секция привязки включена в структуру обмена, то она должна быть расположена после заголовочной секции и перед любой секцией ссылки, данных или подписи. Секция должна начинаться со специальной лексемы "ANCHOR;" и заканчиваться специальной лексемой "ENDSEC;".

Каждая запись в секции привязки должна определять одно внешнее имя. Имя привязки не должно использоваться для любой другой привязки в той же самой структуре обмена. Имя привязки должно соответствовать требованиям 6.5.4. Имя привязки должно быть UUID, если оно соответствует требованиям, определенным в приложении G.

Пример

Структура секции привязки	Комментарий
ANCHOR;	
<82ff3c50-3610-11e5-a2cb-0800200c9a66> = #10;	/* Элемент привязки, определяемый именем (идентификатором) экземпляра объекта, см. 9.2.1 */
<8eae4370-3610-11e5-a2cb-0800200c9a66> = @20;	/* Элемент привязки, определяемый именем (идентификатором) экземпляра значения, см. 9.2.2 */
<9a9ec060-3610-11e5-a2cb-0800200c9a66> = 30;	/* Элемент привязки, определяемый данными простого типа, см. 9.2.3 */

<a3cee4d0-3610-11e5-a2cb-0800200c9a66> = $; /* Элемент привязки, определяемый нулевым значением, см. 9.2.4 */

<bb2ac46e-3610-11e5-a151-feff819cdc9f> = (1.1, 2.1, 3.1); /* Элемент привязки, определяемый списком, см. 9.2.5 */

<bb2ac770-3610-11e5-a151-feff819cdc9f> = <picture.jpg>; /* Элемент привязки, определяемый ресурсом, см. 9.2.6 */

<d1dbb491-dae8-409b-87dd-f21fd5bbb624> = #INCH; /* Элемент привязки, определяемый константой, определение которой выполнено средствами языка EXPRESS, см. 9.2.7 */

<1231ea63-d573-4d50-81f3-8bb0e9c1cbf5> = #10 {ratio:196.73};/* Элемент привязки с меткой, см. 9.2.8 */

ENDSEC;

9.2 Элемент привязки

Каждый ANCHOR должен связать ANCHOR_NAME с ANCHOR_ITEM. Элемент привязки должен описывать, как привязка представлена в структуре обмена. Один и тот же элемент привязки может использоваться в качестве представления для нескольких привязок и может быть нулевым.

Пример

Структура секции привязки. Возможное использование.
< tool_length > = # 100 ; /* ***Первое использование элемента привязки*** # 100 */
< tool_height > = # 100 ; /* ***Второе использование элемента привязки*** # 100 */
<tool_tip_unused> = $; /* ***Экземпляр недоступен в этой версии структуры обмена*** */

Примечание — Если две привязки в двух структурах обмена имеют одно и то же имя, то следует определить — описывают ли они различные аспекты или различные версии одного и того же понятия. Например, создатель инструмента может использовать привязки для одного и того же понятия в семействе элементов инструмента.

9.2.1 Элемент привязки, определенный именем экземпляра объекта

Если элемент привязки определяется именем экземпляра объекта (см. 6.4.4.3), то значением этого элемента привязки является экземпляр объекта, идентифицированный этим именем.

Примечание — Имя экземпляра объекта может быть определено в ссылочной секции, в этом случае значение элемента привязки будет определено в другой структуре обмена.

Пример

Привязка	Возможное использование
<tool_length> = #100;	/* Первое использование экземпляра объекта #100 в качестве элемента привязки */
<tool_height> = #100;	/* Второе использование экземпляра объекта #100 в качестве элемента привязки */

9.2.2 Элемент привязки, определенный именем экземпляра значения

Если элемент привязки определяется именем экземпляра значения (см. 6.4.4.4), то значение этого элемента привязки должно быть значением, обозначенным этим именем.

Примечание — Экземпляр значения определен в ссылочном разделе.

Пример

Привязка	Возможное использование
<length_ratio> = @10;	/* Первое использование экземпляра значения@10 в качестве элемента привязки */
<height_ratio> = @10;	/* Второе использование экземпляра значения@10 в качестве элемента привязки */
<width_ratio> = $;	/* В настоящей версии обменной структуры значение недоступно */

9.2.3 Элемент привязки, определенный простым типом

Если элемент привязки определяется простым типом (см. 6.4), то значение этого элемента привязки должно быть простым типом.

Пример

Привязка	Возможное использование
<low_precision_pi> = 3.142;	/* Элемент привязки, определяемый данными простого типа */
<message> = 'This is a message';	/* Элемент привязки, определяемый данными простого типа */

9.2.4 Элемент привязки, определенный значением null_value ("$")

Если элемент привязки определяется значением **null_value** ("$"), то значение этого элемента привязки должно иметь значение **null_value.**

Примечание — Привязка, определенная **null_value**, может быть простым использованием, которое не может быть выполнено в данной версии структуры обмена, но было выполнено в предыдущей версии или будет выполнено в будущих версиях.

Пример

Привязка	Возможное использование
<tool_tip_unused> = $;	/* В настоящей версии обменной структуры значение недоступно */

9.2.5 Элемент привязки, определенный списковой структурой

Если элемент привязки определяется структурой ANCHOR_ITEM_LIST (см. 7.2), то значение этого элемента привязки должно быть списком**.**

Пример

Привязка	Возможное использование
<identity> = ((1, 0, 0), (0, 1, 0), (0, 0, 1));	/* Элемент привязки, определяемый как матрица идентичности */

9.2.6 Элемент привязки, определенный ресурсом

Если элемент привязки определяется ресурсом (см. 6.5.1), то значение этого элемента привязки должно быть задано как ресурс.

Примечание — Привязки экспортируют информацию. Они не добавляют информацию, так что они могут быть помечены чем-либо, что соответствует требованиям синтаксиса настоящего стандарта.

Пример

Привязка	Возможное использование
<main_image> = <picture.jpg>;	/* Элемент привязки, определяемый изображением */
<sheet_values> = (<work.xls#A3>,<work.xls#A4>,<work.xls#A5>);	/* Элемент привязки, определяемый списком значений электронной таблицы */

9.2.7 Элемент привязки, определенный константой на языке EXPRESS

Если элемент привязки определяется константой на языке EXPRESS (см. 6.4.4.1 и 6.4.4.2), то значение этого элемента привязки должно быть значением или экземпляром, определенным в первой EXPRESS-схеме объекта **file_schema** (см. 8.2.4).

Пример

Привязка	Возможное использование
<pi> = #ARCHIMEDES_CONSTANT_PI;	/* Элемент привязки, заданный константой пи, определенной средствами языка EXPRESS */

<units> = (#IMPERIAL_LENGTH_INCH, #PLANE_ANGLE_RADIAN, #SOLID_ANGLE_STERADIAN);	/* Элемент привязки, заданный списком постоянных объектов, определенных средствами языка EXPRESS */

9.2.8 Элемент привязки с тегами

Тэг должен связывать с привязкой дополнительную информацию. Эта информация должна выходить за рамки EXPRESS-схемы, определяющей структуру обмена (см. 8.2.4).

Примечания

1 Приложения могут использовать тэги для упрощения обработки структуры обмена. Например, тэг может содержать индекс данных, который может использоваться для более быстрого поиска значений.

2 Объект **file_name.name** заголовочной секции может использоваться для указания типов привязок и тэгов в структуре обмена (см. 8.2.3).

Пример

Привязка	Возможное использование
<bedroom> = #10 {link:<bedroom.jpg>};	/* Тэг (метка), используемый для задания источника дополнительной информации об элементе привязки */
<kitchen> = #20 {label:'Price estimate'} {link:<kitchen_cost.xls>};	/* Тэг (метка), используемый для задания стоимости и метки стоимости элемента привязки */

Примечание — Приложение F описывает привязку ECMAScript для доступа к информации тэга.

10 Ссылочная секция

10.1 Структура ссылочной секции

Синтаксис ссылочной секции определен в таблице 3. Ссылочная секция является необязательной. Если ссылочная секция включена в структуру обмена, то она должен быть расположена после заголовочной секции и любой секции привязки и перед любой секцией данных или секции подписи. Ссылочная секция должна начинаться со специальной лексемы "REFERENCE;" и заканчиваться специальной лексемой "ENDSEC;".

Каждая запись в ссылочной секции должна связывать LHS_OCCURRENCE_NAME с RESOURCE. Если имя вхождения является ENTITY_INSTANCE_NAME, то ресурс должен идентифицировать объект. Если имя вхождения является VALUE_INSTANCE_NAME, то ресурс должен определить значение.

Каждое имя вхождения должно иметь только одну ассоциацию и не должно быть определено в секциях данных структуры обмена.

Пример

Структура ссылочной секции	Комментарий
REFERENCE;	
#10 = <gost.e-ecolog.ru stp#shape>;	/* Ссылка на URI с обозначением фрагмента ресурса, см. 10.2 */
#20 = <building_698.ifc>;	/* Ссылка на URI без обозначения фрагмента ресурса, см. 10.2.1 */
#30 = <#10b9903b-fe1a-41dd-91e2-dd8bf06f262d>;	/* Ссылка на заданный посредством UUID идентификатор фрагмента, см. 10.2.2 */
#40 = <#wheel>;	/* Ссылка на идентификатор фрагмента, определяемого в том же файле, см. 10.2.3 */
#50 = <gost.e-ecolog.ru jt#shape>;	/* Ссылка на файл иного формата, см. 10.2.4 */

```
#60 = <gost.e-ecolog.ru                    /* Ссылка на экземпляр объекта, см. 10.2.5 */
ty>;
@70 = <gost.e-ecolog.ru                    /* Ссылка на значение, см. 10.2.6 */
gost.e-ecolog.ru
#80 = <gost.e-ecolog.ru .                  /* Ссылка, в которой используется имя (идентификатор)
stp#100>;                                  экземпляра объекта версии 1 или версии 2, см. 10.2.7 */
ENDSEC;
```

10.2 Ссылка на URI

Ссылка должна разрешаться к значению или экземпляру объекта в структуре обмена, определенной в настоящем стандарте, к сжатому архиву, удовлетворяющему требованиям, определенным в А.4, приложение А, или к каталогу, имеющему форму несжатого архива, определенную в А.5, приложение А.

Доставленный ресурс должен содержать привязку с тем же именем, что и URI_FRAGMENT_IDENTIFIER URI. ANCHOR_ITEM, определенный этой привязкой, являются результатом ссылки. Определенный ANCHOR_ITEM должен отвечать требованиям к объекту или экземпляру значения, принадлежащему к **file_schema** данной структуры обмена.

Если ресурс распознается в привязке, определенной другим URI, то процесс повторяется.

Если разрешение URI не соответствует этим требованиям, то результатом ссылки будет **null_value** (“$”).

Пример

Ссылка	Комментарий
#1234 = <gost.e-ecolog.ru item>;	/* Ссылка на отдельный инструмент в инструменте в сборе с оправкой */
#1235 = <building_698.ifc#7c9e6679-7425-40de-944b-e07fc1f90ae7>;	/* Ссылка на изделие с указанным UUID (см. приложение G) */
@1236 = <gost.e-ecolog.ru >;	/* Ссылка на значение, определяемое в другой обменной структуре */

Примечание — Циклические ссылки могут быть созданы путем ссылки на значение во внешней структуре обмена, которое ссылается на текущую ссылку в текущей структуре обмена. Эти ссылки недопустимы и система обработки должна присвоить им значение NULL.

10.2.1 Ссылка на URI без идентификатора фрагмента

Если ресурс не включает URI_FRAGMENT_IDENTIFIER, то ссылке должно быть присвоено значение **null_value** ("$").

Пример

Ссылка	Комментарий
#10 = <gost.e-ecolog.ru >;	/* результатом является нулевое значение (null_value) ("$") */

10.2.2 Ссылка на URI с идентификатором фрагмента, определенным UUID

Если ресурс, определенный URI, привязан универсальным уникальным идентификатором (UUID) (см. приложение G), то идентификатор должен быть преобразован в объект или значение, привязанное этим уникальным идентификатором.

Если ресурс имеет более одного URI_FRAGMENT_IDENTIFIER, то структура обмена, идентифицированная URI, должна иметь привязку с этим UUID.

Если ресурс определяется одним URI_FRAGMENT_IDENTIFIER, то система обработки должна отвечать за поиск сервиса, который может идентифицировать структуру обмена, содержащую UUID.

Примечание — Например, он может обратиться в службу реестра, которая может быть PLM или аналогичной системой, чтобы запросить структуру обмена.

Пример

Ссылка	Комментарий
#123 = <gost.e-ecolog.ru gost.e-ecolog.ru >;	пляр объекта, который должен находиться в файле, возвращаемом сервером по URI*/
#123 = <#97c6e1f0-3544-11e5-a2cb-0800200c9a66>;	/* раскрывается как привязываемый посредством UUID экземпляр объекта, который должен находиться в обменной структуре, возвращаемой службой регистрации */
@abc = <cutter.stp#825cd420-3547-11e5-a2cb-0800200c9a66>;	/* раскрывается как привязываемое посредством UUID значение, которое должно находиться в файле, обозначаемом посредством URI */

10.2.3 Ссылка на URI, который является только идентификатором фрагмента и не является UUID

Если ресурс определен одним URI_FRAGMENT_IDENTIFIER и этот идентификатор фрагмента не является UUID, то идентификатору должно быть разрешено быть привязкой с тем же именем в текущей структуре обмена.

Пример

Ссылка	Комментарий
#123 = <#wheel>;	/* раскрывается как экземпляр объекта, определяемый местным пунктом привязки */
@124 = <#size>;	/* раскрывается как значение, определяемое местным пунктом привязки */

10.2.4 Ссылка на структуру обмена в другом формате

Если ресурс идентифицирует файл в другом формате, то сервер, идентифицированный URI, должен преобразовать файл в соответствии с требованиями этой структуры обмена.

Пример

Ссылка	Комментарий
#100 = <gost.e-ecolog.ru >;	/* Ссылка на файл формата 3DXML */
#200 = <gost.e-ecolog.ru >;	/* Ссылка на файл формата JT */
@300 = <gost.e-ecolog.ru >;	/* Ссылка на электронную таблицу формата XLS */

10.2.5 Ссылка на фрагмент URI, разрешающий экземпляр объекта

Если идентификатор фрагмента URI разрешается экземпляру объекта, то LHS_OCCURRENCE_NAME должно быть именем экземпляра объекта.

Пример

Ссылка	Комментарий
#100 = <gost.e-ecolog.ru >;	/* Ссылка на экземпляр объекта, определяемый в секции привязки файла hub.stp */
#200 = <gost.e-ecolog.ru >;	/* Ссылка на экземпляр объекта, определяемый в секции привязки файла STEP, полученного в результате трансляции файла axle.jt */

10.2.6 Ссылка на фрагмент URI, который разрешается в значение

Если идентификатор фрагмента URI разрешается значению, то LHS_OCCURRENCE_NAME должно быть именем экземпляра значения.

Пример

Ссылка		Комментарий
@100 = <gost.e-ecolog.ru	>;	/* Ссылка на значение, определяемое в секции привязки файла hub.stp */
@200 = <gost.e-ecolog.ru	>;	/* Ссылка на значение, определяемое в секции привязки файла STEP, полученного в результате трансляции файла axle.jt */

Примечание — Значение может быть простого типа или списочная структура.

10.2.7 Ссылка, использующая фрагмент URI, который является именем экземпляра объекта

Если идентификатор фрагмента URI является числом без знака, которое соответствует требованиям имени экземпляра объекта, то он будет разрешен экземпляру объекта с этим числом в ссылочной структуре обмена и **null_value** ("$") в противном случае.

Примечание — Ссылки на имена экземпляров разрешены, так что структуры обмена издания 3 могут ссылаться на данные издания 2 настоящего стандарта.

11 Секции данных

11.1 Структура секции данных

Секция данных содержит экземпляры, передаваемые через структуру обмена. Каждая секция данных содержит экземпляры объектов, соответствующие одной EXPRESS-схеме, определенной в заголовочной секции.

Синтаксис секции данных представлен в таблице 3. Каждая секция данных должна начинаться с ключевого слова "DATA". Если в структуру обмена включено несколько секций данных, в каждой из них за ключевым словом "DATA" должен следовать список параметров (PARAMETER_LIST), содержащий строку (STRING) или список (LIST).

Первым параметром должна быть строка (STRING), содержащая индивидуальное имя секции. Вторым параметром должен быть список (LIST), содержащий одну строку (STRING). Данная строка должна содержать имя схемы, управляющей заданной секцией данных. Имя данной схемы должно входить в заголовочную секцию объекта **file_schema**.

Если структура обмена содержит только одну секцию данных, тогда список параметров (PARAMETER_LIST) может быть опущен. В этом случае в заголовочной секции объекта **file_schema** должна быть определена только одна схема, управляющая заданной секцией данных.

Каждая секция данных должна заканчиваться специальной лексемой "ENDSEC;".

Примечания

1 Раздел данных является необязательным в настоящем стандарте, так что структуры обмена могут быть созданы для определения ссылок на пересылку в другие структуры.

2 В приложении H представлен полный пример секции данных в структуре обмена.

11.2 Экземпляры объектов секции данных

Каждый экземпляр объекта, который не определяется внешней ссылкой (см. раздел 10), должен отображаться на конструкцию ENTITY_INSTANCE (см. таблицу 3) в секции данных, как определено в 12.2. Каждый экземпляр объекта должен быть представлен в секции данных не более одного раза: различающиеся экземпляры объектов должны иметь различные имена. Экземпляры объектов в структуре обмена не нуждаются в упорядочении. Ссылка на имя экземпляра объекта может появляться до того, как оно будет определено в структуре обмена.

11.3 Экземпляры объектов секции данных, определяемые пользователем

Экземпляр объекта, определяемый пользователем, не входит в EXPRESS-схему, определенную в заголовочной секции. Экземпляры объекта, определяемые пользователем, должны соответствовать синтаксису всех экземпляров объектов этой секции данных, исключая выбор USER_DEFINED_KEYWORD, который должен быть использован в SIMPLE_RECORD, входящем в данное описание. Смысловое содержание экземпляров объектов, определяемых пользователем, их количество, типы данных и содержание соответствующих атрибутов являются предметом соглашения между сторонами, использующими заданную структуру обмена.

Пример

```
DATA;
   .
   .
   .
#1=PT(1.0,2.0,3.0); ----------------------> ЭКЗЕМПЛЯР ОБЫЧНОГО ОБЪЕКТА
#2=PT(1.0,2.0,5.0);
   .
   .
   .
#12=!MYCURVE(0.0,0.0,0.0,1.0,$,$,$); ----> ЭКЗЕМПЛЯР ОБЪЕКТА, ОПРЕДЕЛЕННОГО
ПОЛЬЗОВАТЕЛЕМ
   .
   .
   .
ENDSEC;
```

Примечание — До использования синтаксиса, определенного пользователем в соответствии с настоящим разделом, следует описать EXPRESS-схему, определяющую информацию, задаваемую пользователем, и ее кодирование в отдельной секции данных.

12 Отображение из EXPRESS в структуру обмена

12.1 Отображение типов данных EXPRESS

Данный раздел описывает, каким образом экземпляры типов данных, определенных в языке EXPRESS, описанном в ИСО 10303-11, отображаются в структуру обмена.

Язык EXPRESS включает в себя объявления TYPE (типов), ENTITY (объектов) и CONSTANT (констант), спецификации ограничений и описание алгоритмов. Только экземпляры типов данных, определенные как типы данных EXPRESS с помощью объявлений TYPE и ENTITY, отображаются в структуру обмена. Другие элементы языка в структуру обмена не отображаются (см. таблицу 5).

Таблица 5 — Краткая справочная таблица отображений

Элемент EXPRESS	Отображается в:
ARRAY	список (list)
BAG	список (list)
BOOLEAN	булевскую переменную (boolean)
BINARY	двоичное (binary)
CONSTANT	экземпляр объекта или значения
DERIVED ATTRIBUTE	НЕ ОТОБРАЖАЕТСЯ
ENTITY	экземпляр объекта

Окончание таблицы 5

Элемент EXPRESS	Отображается в:
ENTITY AS ATTRIBUTE	имя (идентификатор) экземпляра объекта
ENUMERATION	перечисление (enumeration)
FUNCTION	НЕ ОТОБРАЖАЕТСЯ
INTEGER	целое (integer)
INVERSE	НЕ ОТОБРАЖАЕТСЯ
LIST	список (list)
LOGICAL	перечисление (enumeration)
NUMBER	вещественное (real)
PROCEDURE	НЕ ОТОБРАЖАЕТСЯ
REAL	вещественное (real)
REMARKS	НЕ ОТОБРАЖАЕТСЯ
RULE	НЕ ОТОБРАЖАЕТСЯ
SCHEMA	НЕ ОТОБРАЖАЕТСЯ
SELECT	См. 12.1.8
SET	список (list)
STRING	строку (string)
TYPE	См.12.1.6
UNIQUE rule	НЕ ОТОБРАЖАЕТСЯ
WHERE RULES	НЕ ОТОБРАЖАЕТСЯ

12.1.1 Отображение простых типов данных EXPRESS

12.1.1.1 Тип данных **integer** (целое)

Значения данных в EXPRESS типа INTEGER отображаются в структуру обмена как целочисленный тип данных (см. 6.4.1), как имена значений констант (см. 6.4.4.2) или как имена значений экземпляров (см. 6.4.4.4).

Примечание — В настоящем стандарте значения экземпляра могут быть определены в ссылочном разделе (см. раздел 10) и в виде констант EXPRESS.

12.1.1.2 Тип данных **string** (строковый)

Значения данных в EXPRESS типа STRING отображаются в структуру обмена как строковый тип данных (см. 6.4.3), как имена значений констант (см. 6.4.4.2) или как имена значений экземпляров (см. 6.4.4.4).

12.1.1.3 Тип данных **boolean** (булевский)

Значения данных в EXPRESS типа BOOLEAN отображаются в структуру обмена как данные перечисляемого типа (см. 6.4.5), как имена значений констант (см. 6.4.4.2) или как имена значений экземпляров (см. 6.4.4.4).

Данные в EXPRESS типа BOOLEAN должны быть обработаны как предопределенный перечисляемый тип данных со значением, кодированным графическим символом "T" или "F". Эти значения соответствуют true (истина) и false (ложь).

12.1.1.4 Тип данных **logical** (логический)

Значения данных в EXPRESS типа LOGICAL отображаются в структуру обмена как данные перечисляемого типа (см. 6.4.5), как имена значений констант (см. 6.4.4.2) или как имена значений экземпляров (см. 6.4.4.4).

Данные в EXPRESS типа LOGICAL должны быть обработаны как предопределенный перечисляемый тип данных со значением, кодированным графическими символами "T", "F" или "U". Эти значения соответствуют true (истина), false (ложь) и unknown (неизвестно).

12.1.1.5 Тип данных **real** (вещественный)

Значения данных в EXPRESS типа REAL должны быть отображены в структуру обмена как вещественный тип данных (см. 6.4.2), как имена значений констант (см. 6.4.4.2) или как имена значений экземпляров (см. 6.4.4.4).

Пример — Определение объекта в языке EXPRESS.

```
ENTITY widget;
  i1:  INTEGER;   ----------> A
  i2:  INTEGER;   ----------> B
  s1: STRING(3);  ----------> C
  s2: STRING;     ----------> D
  l:   LOGICAL;   ----------> E
  b:   BOOLEAN;   ----------> F
  r1:  REAL(4);   ----------> G
  r2:  REAL;      ----------> H
  r3:  REAL;      ----------> L
  r4:  REAL;      ----------> M
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#2 = WIDGET(99, 99999, 'ABC', 'ABCDEFG'.T., .F., 9., 1.2345, @10, @PI);
                .  ^    ^     ^       ^       ^   ^   ^   ^     ^     ^
                   |    |     |       |       |   |   |   |     |     |
                   |    |     |       |       |   |   |   |     |     |
                   A    B     C       D       E   F   G   H     L     M
```

A: в этом экземпляре объекта атрибут i1 имеет значение 99;

B: в этом экземпляре объекта атрибут i2 имеет значение 99999;

C: в этом экземпляре объекта атрибут si имеет значение 'ABC'. Это значение попадает в диапазон (три символа), определенный для данного атрибута;

D: в этом экземпляре объекта атрибут s2 имеет значение 'ABCDEFG';

E: в этом экземпляре объекта атрибут l имеет значение TRUE (истина);

F: в этом экземпляре объекта атрибут b имеет значение FALSE (ложь);

G: в этом экземпляре объекта атрибут r1 имеет значение 9. Требование точности не влияет на кодирование;

H: в этом экземпляре объекта атрибут r2 имеет значение 1.2345;

L: в этом экземпляре объекта атрибут r3 имеет значение, определенное в ссылочном разделе;

M: в этом экземпляре объекта атрибут r4 имеет значение, определенное в EXPRESS-схеме.

12.1.1.6 Тип данных **binary** (двоичный)

Значения данных в EXPRESS двоичного типа отображают в структуру обмена как двоичный тип данных (см. 6.4.6), как имена значений констант (см. 6.4.4.2) или как имена значений экземпляров (см. 6.4.4.4).

Пример — Определение объекта в языке EXPRESS.

```
ENTITY picture;
  bn :  BINARY;
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#4 = PICTURE("1556FB0");
                    ^
                    |
                    A
```

A: в этом экземпляре атрибут bn закодирован как "1556FB0", что соответствует последовательности битов 101 0101 0110 1111 1011 0000.

12.1.1.7 Тип данных **number** (числовой)

Значения данных в EXPRESS типа NUMBER должны быть отображены в структуру обмена как вещественный тип данных. Формирование вещественного типа данных описано в 6.4.2.

12.1.2 Тип данных list (список)

Значения данных в EXPRESS типа LIST отображают в структуру обмена как данные спискового типа. В разделе 7 описан состав данных спискового типа. Если список пустой, то он должен быть закодирован как *левая скобка* "(", за которой сразу следует *правая скобка* ")". Внутри списка каждый экземпляр типа элемента должен быть закодирован, как требуется (согласно разделу 12) для каждого типа данных в EXPRESS.

Значения данных в EXPRESS типа LIST могут быть определены в ссылочной секции. Значение сопоставляется с использованием имени значения экземпляра, определенного для этого экземпляра в ссылочной секции. Значение экземпляра должно быть совместимо с типом EXPRESS.

Значения данных в EXPRESS типа LIST могут быть определены константами EXPRESS. Значение должно быть сопоставлено с использованием значения, определенного для этого экземпляра в схеме EXPRESS. Значение экземпляра должно быть совместимо с типом EXPRESS.

Примечание — Если в конкретном экземпляре объекта отсутствует значение для необязательного (OPTIONAL) атрибута, чей тип данных — LIST, то атрибут должен быть закодирован *знаком доллара* ("$"), как определено в 12.2.2, и это не пустой список.

Пример — Определение объекта в языке EXPRESS.

```
ENTITY widget;
  attribute1:  LIST [0 : ?] OF INTEGER; ------------------> A
  attribute2:  LIST [1 : ?] OF INTEGER; ------------------> B
  attribute3:  OPTIONAL LIST [1 : ?] OF INTEGER; --> C
  attribute4:  REAL; ---------------------------------------> D
  attribute5:  LIST [1 : ?] OF INTEGER; ------------------> E
END_ENTITY;
Образец экземпляра объекта в секции данных:
#4 = WIDGET((), (1,2,@10), $, 2.56, @PI);
                 ^      ^      ^   ^     ^
                 |      |      |   |     |
                 A      B      C   D     E
```

A: атрибут attribute1 является пустым списком (список, содержащий ноль элементов);

B: в этом экземпляре атрибут attribute2 содержит три элемента. Один из экземпляров определяется экземпляром @10 в ссылочной секции;

C: в этом экземпляре атрибут attribute3 не имеет значения;

D: в этом экземпляре атрибут attribute4 имеет значение 2.56;

E: в этом экземпляре атрибут attribute5 имеет значение LIST, определенное в EXPRESS константой PI.

12.1.3 Тип данных array (массив)

Значения данных в языке EXPRESS типа ARRAY должны быть отображены в структуру обмена как данные спискового типа. Раздел 7 описывает состав данных спискового типа. Если атрибут в языке EXPRESS является многомерным массивом, он должен быть закодирован как список списков, вложенный на глубину, равную размерности массива. В создании таких списков самый внутренний список, содержащий только экземпляры типа элемента, должен соответствовать самому правому из определений ARRAY в операторе языка EXPRESS, определяющем объект. Упорядочение элементов при кодировании осуществляется так, что для каждого элемента следующего внешнего списка кодируются элементы внутреннего списка. Данный порядок означает, что в каждом списке самый правый индекс будет изменяться первым. Внутри списка каждый экземпляр типа элемента должен быть закодирован, как требуется (согласно разделу 12) для каждого типа данных в EXPRESS. Если тип данных массива имеет необязательные (OPTIONAL) элементы, то любой элемент, для которого отсутствует значение, должен быть закодирован знаком доллара ("$").

Значения данных в языке EXPRESS типа ARRAY могут быть определены в ссылочной секции. Значение должно отображаться с использованием имени экземпляра значения, определенного для этого экземпляра в ссылочной секции. Значение экземпляра должно быть совместимо с типом EXPRESS.

Значения данных в языке EXPRESS типа могут быть определены константами EXPRESS. Значение должно отображаться с использованием значения, определенного для этого экземпляра в схеме EXPRESS. Значение экземпляра должно быть совместимо с типом EXPRESS.

Примечание — Если в конкретном экземпляре объекта отсутствует значение для необязательного (OPTIONAL) атрибута, тип данных которого ARRAY, этот атрибут должен быть закодирован *знаком доллара* ("$") как определено в 12.2.2, и это не пустой список.

Примеры

1 Определение объекта в языке EXPRESS.

```
X : ARRAY[1:5] OF ARRAY[100:102] OF INTEGER
Это кодируется в следующем порядке:
( (X [1,100], X [1,101], X [1,102] ),
  (X [2,100], X [2,101], X [2,102] ),
  (X [3,100], X [3,101], X [3,102] ),
  (X [4,100], X [4,101], X [4,102] ),
  (X [5,100], X [5,101], X [5,102] )  )
```

2 Определение объекта в языке EXPRESS.

```
ENTITY widget;
  attribute1:  ARRAY [-1 : 3] OF INTEGER;     ----------------------------> A
  attribute2:  ARRAY [1 : 5] OF OPTIONAL INTEGER;     -------------> B
  attribute3:  ARRAY [1 : 2] OF ARRAY [1 : 3] OF INTEGER;     ------> C
END_ENTITY;
Образец экземпляра объекта в секции данных:
#30 = WIDGET((1,2,3,4,5) , (1,2,3,$,5) , ((1,2,3),(4,5,6)));
                          ^           ^             ^
                          |           |             |
                          A           B             C
A: атрибут attribute1 содержит следующие значения:
attribute1 [-1] = 1
attribute1 [0]  = 2
attribute1 [1]  = 3
attribute1 [2]  = 4
attribute1 [3]  = 5
B: атрибут attribute2 содержит следующие значения:
attribute2 [1] = 1
attribute2 [2] = 2
attribute2 [3] = 3
attribute2 [4] = отсутствует
attribute2 [5] = 5
Пропуск значения в массиве предусмотрен в определении EXPRESS-схемы.
C: атрибут attribute3 содержит следующие значения:
attribute3 [1,1] = 1
attribute3 [1,2] = 2
attribute3 [1,3] = 3
attribute3 [2,1] = 4
attribute3 [2,2] = 5
attribute3 [2,3] = 6
```

12.1.4 Тип данных set (набор)

Значения данных в языке EXPRESS типа SET должны быть отображены в структуру обмена как данные спискового типа. Раздел 7 описывает состав данных спискового типа. Внутри списка каждый экземпляр типа элемента должен быть закодирован, как требуется (согласно разделу 12) для каждого типа данных в EXPRESS. Если SET пустой, список должен кодироваться как *левая скобка* "(", за которой сразу следует *правая скобка* ")".

Значения данных в языке EXPRESS типа SET могут быть определены в ссылочной секции. Значение отображается с использованием имени экземпляра значения, определенного для этого экземпляра в ссылочной секции. Значение экземпляра должно быть совместимо с типом EXPRESS.

Значения данных в языке EXPRESS типа SET могут быть определены константами EXPRESS. Значение отображается с использованием значения, определенного для этого экземпляра в схеме EXPRESS. Значение экземпляра должно быть совместимо с типом EXPRESS.

Примечание — Если в конкретном экземпляре объекта отсутствует значение для необязательного (OPTIONAL) атрибута, чьим типом данных является SET, этот атрибут должен быть закодирован *знаком доллара* ("$"), как определено в 12.2.2, и это не пустой список.

Пример — Определение объекта в языке EXPRESS.

```
ENTITY widget;
  a_number: SET OF INTEGER;
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#2 = WIDGET((0,1,2)); --------> A
#3 = WIDGET((0,$,2)); --------> B
#4 = WIDGET((0,0,2)); --------> C
```

A: в этом экземпляре атрибут a_number определен набором чисел 0, 1, 2;

B: синтаксически экземпляр правилен. Однако экземпляр неправилен относительно определения SET в EXPRESS, поскольку SET не может иметь пропущенных элементов;

C: синтаксически экземпляр правилен. Однако экземпляр неправилен относительно определения SET в EXPRESS, поскольку SET не может иметь одинаковых элементов.

12.1.5 Тип данных bag (мультимножество)

Значения данных в языке EXPRESS типа BAG должны быть отображены в структуру обмена как данные спискового типа. Раздел 7 описывает состав данных спискового типа. Внутри списка каждый экземпляр типа элемента должен быть закодирован, как требуется (согласно разделу 12) для каждого типа данных в EXPRESS. Если BAG пустое, список должен кодироваться как *левая скобка* "(", за которой сразу следует *правая скобка* ")".

Значения данных в языке EXPRESS типа BAG могут быть определены в справочной секции. Значения должны быть отображены с использованием имени экземпляра значения, определенного для этого экземпляра в ссылочной секции. Значение экземпляра должно быть совместимо с типом EXPRESS.

Значения данных в языке EXPRESS типа BAG могут быть определены константами EXPRESS. Значения должны быть отображены с использованием значения, определенного для этого экземпляра в схеме EXPRESS. Значение экземпляра должно быть совместимо с типом EXPRESS.

Примечание — Если в конкретном экземпляре объекта отсутствует значение для необязательного (OPTIONAL) атрибута, чьим типом данных является BAG, атрибут должен быть закодирован *знаком доллара* ("$"), как определено в 12.2.2, и это не пустой список.

Пример — Определение объекта в языке EXPRESS.

```
ENTITY widget;
  a_numbers: BAG OF INTEGER;
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#2 = WIDGET((0,1,1,2));  --------> A
#3 = WIDGET((0,$,2));    --------> B
```

A: в этом экземпляре атрибут a_numbers определен набором чисел 0, I, 1, 2;

B: синтаксически экземпляр правилен. Однако экземпляр неправилен относительно определения BAG в EXPRESS, поскольку BAG не может иметь пропущенных элементов.

12.1.6 Простые определенные типы

Простой определенный тип является типом, определяемым объявлением EXPRESS-типа, в котором опорным типом не является перечисляемый тип (ENUMERATION) или выбираемый тип (SELECT). Простой определенный тип должен быть отображен в структуру обмена как тот тип данных, который использовался в его определении.

Пример — Определение объекта в языке EXPRESS.

```
TYPE
  type1 =  INTEGER;
END_TYPE;

TYPE
  type2 = LIST [1 : 2] of REAL;
END_TYPE;

ENTITY widget;
  attribute1:  LOGICAL; ----------> A
  attribute2:  TYPE1;   ----------> B
  attribute3:  TYPE2;   ----------> C
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#4 = WIDGET( .T., 256, (1.0,0.0));
                ^    ^       ^
                |    |       |
               A    B       C
```

A: в этом экземпляре значением атрибута attribute1 является TRUE (истина);

B: атрибут type1 является целым типом и, следовательно, значение 256 допустимо;

C: атрибут type2 является списком и, следовательно, список из двух вещественных (REAL) элементов допустим.

12.1.7 Тип данных enumeration (перечисление)

Значения данных в языке EXPRESS типа ENUMERATION должны быть отображены в структуру обмена как перечисляемый тип данных. В 6.4.5 описано содержание перечисляемого типа данных.

Если документ, определяющий конкретную схему, экземплярами которой является предмет этой секции данных, задает набор сокращенных наименований для перечисляемых значений в этой схеме, тогда фактическим значением конкретного экземпляра перечисления (ENUMERATION) может быть сокращенное наименование, соответствующее одному из перечисляемых значений в данной EXPRESS-схеме. В противном случае фактическим значением должно быть одно из перечисляемых значений в EXPRESS-схеме. В любом случае *строчные буквы* должны быть преобразованы в *прописные буквы*, а значение должно быть ограничено отделяющими *точками* ".", как это определено в выводе ENUMERATION в таблице 2.

Пример — Определение объекта в языке EXPRESS.

```
TYPE
  primary_colour = ENUMERATION OF (red, green, blue);
END_TYPE;

ENTITY widget;
  p_colour: primary_colour;    ----------> A
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#2 = WIDGET(.RED.);
              ^
              |
             A
```

A: в этом экземпляре объекта значением атрибута p_colour является RED.

12.1.8 Выбираемый тип данных

Выбираемый тип данных в языке EXPRESS определяет список типов данных, называемый список выбора ("select-list"), значениями которого являются правильные экземпляры данных выбираемого типа. Экземпляр данных выбираемого типа должен быть значением по меньшей мере одного из типов данных в списке выбора. Значение кодируется в структуре обмена в соответствии со следующей процедурой:

- если значение является экземпляром типа данных объекта в списке выбора, то оно должно быть отображено в структуру обмена как имя экземпляра объекта (см. 6.4.4);

- если значение является экземпляром простого определяемого типа в списке выбора, оно должно быть отображено в структуру обмена как TYPED_PARAMETER (см. таблицу 3), в котором KEYWORD должно обозначать простой определяемый тип, как определено ниже, а PARAMETER должен быть кодированием значения простого определяемого типа, как определено в 12.1.6;

- если значение является экземпляром перечисляемого типа данных в списке выбора, оно должно быть отображено в структуру обмена как TYPED_PARAMETER (см. таблицу 3), в котором KEYWORD должно обозначать перечисляемый тип данных, как определено ниже, а PARAMETER должен быть кодированием значения перечисляемого типа данных, как определено в 12.1.7;

- если значение является экземпляром (вложенного) выбираемого типа данных, то оно должно быть отображено в структуру обмена в виде экземпляра выбираемого типа, как указано в данном пункте.

Для экземпляров простых определяемых типов и перечисляемых типов данных KEYWORD должно обозначать тип данных экземпляра. Обозначенный тип данных должен быть одним из типов в списке выбора. Если документ, определяющий схему, экземпляры которой являются предметом секции данных, также определяет набор сокращенных имен простых определяемых и перечисляемых типов в данной схеме, KEYWORD должно быть сокращенным именем, соответствующим типу данных экземпляра. В противном случае KEYWORD должно быть самим именем простого определяемого типа или перечисляемого типа данных. В обоих случаях все строчные

буквы должны быть преобразованы в соответствующие прописные буквы, т. е. кодирование не должно содержать строчных букв.

Примечания

1 Если имеется тип данных (в списке выбора), значениями экземпляров которого является сам выбираемый тип данных, тогда настоящий пункт должен быть использован рекурсивно для кодирования значения. Реально могут быть закодированы только типы данных объекта, простые определяемые типы и перечисляемые типы данных (см. пример 2).

2 В соответствии с ИСО 10303-11 экземпляр подтипа типа данных объекта является экземпляром типа данных объекта. Так, "экземпляр типа данных объекта в списке выбора" включает в себя экземпляры подтипов соответствующих типов данных объекта.

3 Если типы данных объекта в списке выбора не являются взаимоисключающими, тогда значение выбираемого типа данных может быть приписано нескольким типам данных объекта в списке выбора (см. пример 1).

4 Если значение не является экземпляром объекта, то оно является экземпляром простого определяемого типа или перечисляемого типа данных. Однако значение может быть фактическим экземпляром нескольких (вложенных) выбираемых типов данных и, следовательно, приписываться нескольким типам в исходном списке выбора (см. пример 2).

Примеры

1 Определение объекта в языке EXPRESS.

```
ENTITY Leader SUBTYPE OF (Employee);
  project: STRING;
END_ENTITY;

ENTITY Manager SUBTYPE OF (Employee);
  unit: STRING;
END_ENTITY;

ENTITY Employee;
  name: STRING;
END_ENTITY;

TYPE Supervisor = SELECT (Manager, Leader);
END_TYPE;


ENTITY Meeting;
  date:        STRING;
  attendees:    SET [2:?] OF Supervisor;
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#1 = LEADER('J. Brahms','Academic Festival');
#2 = MANAGER('S. Ozawa', 'Tokyo Symphony');
#3 = (EMPLOYEE('G. Verdi') LEADER('Aida') MANAGER('La Scala'));
#4 = MEETING('14921012', (#1, #2, #3));
```

Вторым атрибутом экземпляра #4 являются участники: SET OF Supervisor. Экземпляром #1 является Leader, и поэтому Supervisor верен. Экземпляром #2 является Manager, и поэтому Supervisor верен. Экземпляр #3 является экземпляром двух типов Leader и Manager из списка выбора Supervisor. Все экземпляры отображаются согласно 6.4.4.

2 Определение объекта в языке EXPRESS.

```
TYPE Mass = SELECT (Mass_Spec, Mass_Substitute); END_TYPE;

TYPE Mass_Spec = SELECT (Measured_Mass, Computed_Mass, Estimated_Mass);
END_TYPE;

TYPE Measured_Mass = REAL;
END_TYPE;

TYPE Computed_Mass = Extended_Real;
END_TYPE;
```

```
TYPE Estimated_Mass = REAL;
END_TYPE;

TYPE Mass_Substitute = SELECT(Weight, Estimated_Mass);
END_TYPE;

TYPE Weight = REAL;
END_TYPE;

TYPE Extended_Real = SELECT (FloatingNumber, NotaNumber);
END_TYPE;

TYPE FloatingNumber = REAL(7);
END_TYPE;

TYPE NotaNumber = ENUMERATION OF (plus_infinity,
  minus_infinity, indeterminate, invalid);
END_TYPE;

ENTITY Steel_Bar;
  bar_length: Extended_Real;
  bar_mass:   Mass;
END_ENTITY;
Образец экземпляра объекта в секции данных:
#1 = STEEL_BAR(FLOATINGNUMBER(77.0), MEASURED_MASS(13.25));
#2 = STEEL_BAR(NOTANUMBER(.INDETERMINATE.),
       ESTIMATED_MASS(10.0));
#3 = STEEL_BAR(FLOATINGNUMBER(77.0),
     COMPUTED_MASS(FLOATINGNUMBER(14.77719)));
```

Первый атрибут экземпляра #1 представляет значение Extended_Real, которое является FloatingNumber. Оно отображается (согласно 12.1.8 для выбираемого типа данных Extended_Real) как TYPED_PARAMETER с KEYWORD FLOATINGNUMBER (простой определяемый тип в списке выбора). Значение 77.0 PARAMETER отображается в структуру обмена, согласно 12.1.6 для FloatingNumber, как значение простого типа REAL.

Второй атрибут экземпляра #1 представляет значение Measured_Mass, которое является правильным значением Mass_Spec и поэтому также правильным значением Mass. Оно отображается (посредством рекурсивного применения 12.1.8, поскольку Mass является выбираемым типом данных, и Mass_Spec — выбираемым типом данных) как TYPED_PARAMETER с KEYWORD MEASURED-MASS (простой определяемый тип в списке выбора). Значение 13.25 PARAMETER отображается в структуру обмена, согласно 12.1.6 для Measured_Mass как значение простого типа REAL.

Первый атрибут экземпляра #2 представляет значение Extended-Real, являющееся значением NotaNumber. Оно отображается (согласно 12.1.8 для Extended-Real) как TYPED.PARAMETER с KEYWORD NOTANUMBER (перечисляемый тип в списке выбора). Значение "indeterminate" PARAMETER отображается в структуру обмена согласно 12.1.7 как перечисляемый тип NotaNumber.

Второй атрибут экземпляра #2 представляет значение Estimated_Mass. Оно является правильным значением Mass_Spec, а также правильным значением Mass_Subsitute и поэтому правильным значением Mass. Это значение фактически представлено двумя (выбираемыми) типами в списке выбора Mass. Оно отображается (посредством рекурсивного применения 12.1.8, поскольку Mass является выбираемым типом данных, а Mass_Spec — выбираемым типом данных) как TYPED_PARAMETER с KEYWORD ESTIMATED_MASS (простой определяемый тип в списке-выбора). Значение 10.0 PARAMETER отображается в структуру обмена, согласно 12.1.6 для Estimated_Mass, как значение простого типа REAL.

Первый атрибут экземпляра #3 тот же, что и первый атрибут экземпляра #1.

Второй атрибут экземпляра #3 представляет значение Computed_Mass, которое является правильным значением Mass_Spec и поэтому правильным значением Mass. Оно отображается (посредством рекурсивного применения 12.1.8, поскольку Mass является выбираемым типом данных, и Mass_Spec — выбираемым типом данных) как TYPED_PARAMETER с KEYWORD COMPUTED_MASS

(простой определяемый тип в списке выбора). Значение PARAMETER кодируется, согласно 12.1.6 для значения Computed_Mass, которое является значением Extended-Real. Extended-Real является выбранным типом данных. Согласно 12.1.8 значение Extended-Real кодируется как TYPED.PARAMETER с KEYWORD FLOATINGNUMBER (простой определяемый тип в списке-выбора) и как значение 14.77719 PARAMETER, кодируемое согласно 12.1.6 для FloatingNumber.

12.2 Отображение типов данных объекта из языка EXPRESS

Экземпляр типа данных объекта из EXPRESS должен быть отображен в структуру обмена как ENTITYJNSTANCE.

Как определено в ИСО 10303-11, "экземпляр простого объекта" (simple entity instance) является экземпляром объекта, не являющегося экземпляром подтипа какого-либо типа данных объекта. Все прочие экземпляры объекта называются "экземплярами сложного объекта" (complex entity instances). Экземпляр простого объекта должен быть отображен согласно 12.2.1, а экземпляр сложного объекта — согласно 12.2.5.

Примечание — Экземпляр простого объекта является экземпляром объекта, который полностью описывается единственным объявлением объекта в EXPRESS. Экземпляр сложного объекта является экземпляром, описание которого включает в себя несколько объявлений объекта, даже если только одно из них содержит явные атрибуты. Экземпляр простого объекта может быть экземпляром супертипа, пока последний не является экземпляром какого-либо подтипа, но экземпляр подтипа всегда является "сложным".

Только явные атрибуты объекта в EXPRESS отображаются в структуру обмена. Специальные средства, однако, применяются к необязательным (OPTIONAL) явным атрибутам (см. 12.2.2), явным атрибутам, значениями которых являются экземпляры объекта (см. 12.2.4), и ко всем переобъявлениям явных атрибутов (см. 12.2.6—12.2.8).

Примечание — К одному и тому же атрибуту может быть применено несколько средств.

12.2.1 Отображение экземпляра простого объекта

Экземпляр простого объекта должен быть отображен в структуру обмена как SIMPLE_ENTITY_INSTANSE. Имя типа данных объекта должно быть отображено в KEYWORD для SIMPLE_RECORD, как определено в 12.2.11.

В структуре обмена каждый явный атрибут должен быть отображен непосредственно в PARAMETER для SIMPLE_RECORD. Порядок параметров (PARAMETER) в структуре обмена должен быть тем же, что и порядок соответствующих атрибутов в объявлении объекта в EXPRESS. Первый PARAMETER должен быть значением первого явного атрибута, второй PARAMETER — значением второго явного атрибута и т. д. Если тип данных объекта в EXPRESS не имеет явных атрибутов, то список параметров (PARAMETER_LIST) должен быть пустым.

Форма каждого PARAMETER должна зависеть от типа данных соответствующего атрибута, как определено в 12.1.

Пример — Определение в языке EXPRESS.

```
TYPE
  primary_colour_abbreviation = ENUMERATION OF (r, g, b);
END_TYPE;

ENTITY widget; ------------------------------------------------> A
  attribute1: INTEGER; ------------------------------------------> B
  attribute2: STRING;  ------------------------------------------> C
  attribute3: LOGICAL; ------------------------------------------> D
  attribute4: BOOLEAN; ------------------------------------------> E
  attribute5: REAL; ---------------------------------------------> F
  attribute6: LIST [1 : 2] of LOGICAL;  -------------------------> G
  attribute7: ARRAY [-1:3]  of INTEGER;  ------------------------> H
  attribute8: PRIMARY_COLOUR_ABBREVIATION; -> I
END_ENTITY;
Образец экземпляра объекта в секции данных:
#1 = WIDGET( 1, 'A', .T., .F., 1.0, (.T.,.F.), (1,0,1,2,3), .R.);
       ^     ^  ^  ^   ^   ^      ^           ^        ^
       |     |  |  |   |   |      |           |        |
       A     B  C  D   E   F      G           H        I
```

A: имя объекта widget из EXPRESS отображается в стандартное ключевое слово WIDGET объекта секции данных;

B: в этом экземпляре объекта атрибут attribute1 имеет значение 1;

C: в этом экземпляре объекта атрибут attribute2 имеет значение A;

D: в этом экземпляре объекта атрибут attribute3 имеет значение T (истина);

E: в этом экземпляре объекта атрибут attribute4 имеет значение F (ложь);

F: в этом экземпляре объекта атрибут attribute5 имеет значение 1.0;

G: в этом экземпляре объекта атрибут attribute6 является списком логических переменных. Список значений:

ATTRIBUTE6(1) = T

ATTRIBUTE6(2) = F

H: в этом экземпляре объекта атрибут attribute7 является массивом целых. Список значений:

ATTRIBUTE7(-1) = 1

ATTRIBUTE7(0) = 0

ATTRIBUTE7(1) = 1

ATTRIBUTE7(2) = 2

ATTRIBUTE7(3) = 3

I: в этом экземпляре объекта атрибут attribute8 является перечислением, attribute8 имеет значение R.

12.2.2 Отображен не необязательных (OPTIONAL) явных атрибутов

Явный атрибут, объявленный как OPTIONAL, не обязан иметь значение в заданном экземпляре объекта. Если необязательное значение поставляется в экземпляре объекта, оно должно быть закодировано в соответствии с типом данных атрибута, как определено в 12.1. Когда необязательное значение атрибута отсутствует в экземпляре объекта, оно должно быть закодировано в структуре обмена как *знак доллара* ("$").

Пример — Определение объекта в языке EXPRESS.

```
ENTITY xxx;
  attribute1: REAL;
  attribute2: REAL;
END_ENTITY;


ENTITY yyy;  ---------------------------------> A
  attribute1: OPTIONAL LOGICAL; -----> B
  attribute2: xxx;  ---------------------------> C
  attribute3: xxx;  ---------------------------> D
  attribute4: OPTIONAL INTEGER; -----> E
  attribute5: OPTIONAL REAL;  ---------> F
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#1=XXX(1.0,2.0);
#2=XXX(3.0,4.0);
#3=YYY($,#2,#1,$,$);
      ^  ^ ^  ^  ^ ^
      |  | |  |  | |
      A  B C  D  E F
```

A: имя объекта yyy в EXPRESS отображается в стандартное ключевое слово YYY объекта секции данных;

B: в этом экземпляре объекта атрибут attribute1 не имеет значения;

C: атрибут attribute2 является ссылкой на объект xxx экземпляром объекта #2;

D: атрибут attribute3 является ссылкой на объект xxx экземпляром объекта #1;

E: в этом экземпляре объекта атрибут attribute4 не имеет значения;

F: в этом экземпляре объекта атрибут attribute5 не имеет значения.

12.2.3 Отображение вычисляемых атрибутов

Вычисляемые атрибуты объекта не должны отображаться в структуру обмена. Когда вычисляемый атрибут в подтипе переобъявляется как атрибут в супертипе, отображение должно происходить, как описано в 12.2.6.

Пример — Определение объекта в языке EXPRESS:

```
ENTITY yyy;
```

```
  q0: Real;
  q1: Real;
  q2: Real;
END_ENTITY;

ENTITY xxx;   ----------------------> A
  p0: yyy;    ----------------------> B
  p1: yyy;    ----------------------> C
  p2: yyy;    ----------------------> D
DERIVE
  attrib5 : vector := func_normal (p0,p1,p2);  ----> E
  attrib4 : real   := func_diameter (p0,p1,p2);  ---> F
END_ENTITY;
Образец экземпляра объекта в секции данных:
#9   = YYY( 0.0, 0.0, 0.0);
#10 = YYY( 1.0, 2.0, 3.0);
#11 = YYY( 4.0, 5.0, 6.0);
#12 = XXX( #9, #10, #11);
          ^    ^    ^    ^
          |    |    |    |
          A    B    C    D
```

A: имя объекта xxx в EXPRESS отображается в стандартное ключевое слово XXX объекта секции данных;

B: p0 является ссылкой на объект yyy экземпляром объекта #9;

C: p1 является ссылкой на объект yyy экземпляром объекта #10;

D: p2 является ссылкой на объект yyy экземпляром объекта #11;

E: атрибут attrib5 в этом экземпляре объекта не отображается, поскольку он является вычисляемым атрибутом;

F: атрибут attrib4 в этом экземпляре объекта не отображается, поскольку он является вычисляемым атрибутом.

12.2.4 Отображение атрибутов, значения которых являются экземплярами объектов

Если экземпляр объекта определен как атрибут другого (ссылающегося на него) экземпляра объекта, первый (на который ссылаются) экземпляр объекта должен быть отображен в структуру обмена как имя экземпляра объекта (см. 6.4.4).

Ссылка на этот экземпляр объекта должен быть определена в EXPRESS-схеме, ссылочной секции или одной из секций данных, т. е. где-то в EXPRESS-схеме, ссылочной секции или секции данных ссылочный экземпляр объекта должен находиться слева от *знака равенства* "=". Это определение может предшествовать применению экземпляра объекта в качестве атрибута или следовать за ним. Его описание не должно входить в эту секцию в качестве атрибута используемого экземпляра объекта.

Примечания

1 В соответствии с 6.4.4.1, если ссылочный экземпляр определяется константой EXPRESS, то первым символом его имени будет прописная буква (UPPER).

2 В приложении E описаны методы оценки соответствия схемы при наличии в структуре обмена нескольких секций данных, основанных на разных EXPRESS-схемах, включая ссылки между экземплярами объектов, определенных в этих секциях на основе разных EXPRESS-схем.

Пример — Определение объекта в языке EXPRESS.

```
ENTITY yyy;
  x : REAL;
  y : REAL;
  z : REAL;
END_ENTITY;

ENTITY xxx;
  p0 :  yyy;  ---------------> A
  p1 :  yyy;  ---------------> B
END_ENTITY;
Образец экземпляра объекта в секции данных:
#1 = YYY(3., 4., 5.);
#2 = XXX (#1, #3);
#3 = YYY (1., 2., 3.);
```

12.2.5 Объекты, определенные как подтипы других объектов

ИСО 10303-11 определяет экземпляры объекта, имеющего раздел SUBTYPE (подтип), являющийся "экземплярами сложных объектов", так, что они могут включать в себя атрибуты из нескольких объявлений типов объектов. В настоящем пункте определено, как экземпляры сложных объектов должны быть отображены в структуру обмена.

Экземпляры сложного объекта должны быть отображены в структуру обмена на основе одного из двух правил: внутреннего отображения или внешнего отображения. К каждому экземпляру подтипа объекта должно быть применено одно правило отображения. Выбор правила отображения для каждого экземпляра объекта представлены в 12.2.5.1.

Примечания

1 Выбор отображения зависит в большей мере от экземпляра объекта, чем от его типа. Для разных экземпляров одного и того же типа данных объекта возможно использование разных отображений в зависимости от того, являются ли они экземплярами подтипов и какие подтипы они представляют.

2 Настоящий пункт применим только к экземплярам сложного объекта. Нет необходимости применять его к каждому экземпляру объекта супертипа. В частности, он не применяется к экземпляру супертипа, который не является экземпляром любого подтипа. Такие экземпляры могут существовать, если супертип не является абстрактным супертипом и подтипом какого-либо другого объекта. Такие экземпляры отображают согласно 12.2.1.

12.2.5.1 Выбор отображения

Набор определений типа данных объекта, которые связаны выражениями подтипа и явного или неявного супертипа, определяет набор структур экземпляров сложного объекта, на который ссылаются как на определяемое множество в приложении В ИСО 10303-11. Каждый член определяемого множества устанавливает список имен типов данных объектов.

Каждый конкретный экземпляр типа данных объекта соответствует одному элементу определяемого множества. Отображение, применяемое к конкретному экземпляру, зависит от члена определяемого множества, которому соответствует экземпляр.

Для того чтобы установить, какое из правил отображения надо применить к данному экземпляру объекта:

a) определяют список имен типов данных объекта, который становится элементом определяемого множества, соответствующим экземпляру объекта;

b) отбирают из списка все типы объектов, не имеющие подтипы, и все типы объектов, которые могут иметь подтипы, но для которых не определены подтипы в списке (члене определяемого множества) для данного экземпляра;

c) в случае определения только одного типа данных объекта его следует считать "конечным типом данных объекта" (leaf entity data type), и должно быть применено внутреннее отображение. В противном случае должно быть использовано внешнее отображение.

Примечание — При реализации положения перечисления b) должен быть отобран по меньшей мере один тип объекта.

12.2.5.2 Внутреннее отображение

Если используется внутреннее отображение, то экземпляр объекта должен быть отображен в SIMPLE_ENTITY_1NSTANCE (см. таблицу 3). Ключевое слово (KEYWORD) должно быть именем конечного типа данных объекта, как указано в 12.2.11. Список параметров (PARAMETER_LIST) должен содержать значения унаследованных явных атрибутов всех объектов супертипа и явных атрибутов конечного типа данных объекта. Порядок, в котором унаследованные и явные атрибуты будут появляться в структуре обмена, должен быть определен следующим образом:

- все унаследованные атрибуты должны появляться последовательно перед явными атрибутами любого объекта:
- атрибуты объекта супертипа должны наследоваться в порядке их появления в самом объекте супертипа;
- если объект супертипа сам является подтипом другого объекта, то атрибуты более высокого супертипа должны наследоваться первыми;
- когда указано несколько объектов супертипа, атрибуты объектов супертипа должны быть обработаны в порядке, определенном в выражении SUBTYPE OF.

В результате этой процедуры на объект супертипа может быть несколько ссылок. В этом случае все ссылки, кроме первой, должны быть игнорированы.

Примеры

1 Простое отношение подтип/супертип. Определение объекта в языке EXPRESS.

```
ENTITY aa ABSTRACT SUPERTYPE OF (ONEOF(bb,cc));  ---> A
  attrib_a: zz;  ------------------------------------------------------------> B
END_ENTITY;

ENTITY bb SUBTYPE OF (aa)
       ABSTRACT SUPERTYPE OF (ONEOF(xx)); ----------------> C
  attrib_b1: yy;  -----------------------------------------------------------> D
  attrib_b2: yy;  -----------------------------------------------------------> E
END_ENTITY;

ENTITY cc SUBTYPE OF (aa);
  attrib_c : REAL;
END_ENTITY;

ENTITY xx SUBTYPE OF (bb);
  attrib_x:  REAL;  ---------------------------------------------------------> F
END_ENTITY;

ENTITY zz;
  attrib_z : STRING;
END_ENTITY;

ENTITY yy;
  attrib_1 : REAL;
  attrib_2 : REAL;
  attrib_3 : REAL;
END_ENTITY;
```

Образец экземпляра объекта типа данных объекта xx в секции данных:

```
#1 = ZZ('ZATTR');
#2 = YY(1.0, 2.0, 0.0);
#3 = YY(2.0, 2.0, 0.0);

#4 = XX(#1, #2, #3, 4.0);
          ^    ^    ^    ^
          |    |    |    |
          B    D    E    F
```

A: поскольку объект aa является абстрактным супертипом, он не отображается в структуру обмена;

B: атрибут attrib_a будет отображен в секции данных как унаследованный атрибут того объекта, который прямо или косвенно является подтипом объекта aa. В этом случае атрибут attrib_a представлен первым атрибутом экземпляра объекта xx и ссылается на zz экземпляр объекта #1;

C: поскольку объект bb является абстрактным супертипом, он не отображается в структуре обмена;

D: атрибут attrib_b1 будет отображен в секции данных как унаследованный атрибут объекта, который прямо или косвенно является подтипом объекта bb. В этом случае атрибут attrib_b1 представлен вторым атрибутом экземпляра объекта xx и ссылается на yy — экземпляр объекта #2;

E: атрибут attrib_b2 будет отображен в секции данных как унаследованный атрибут объекта, который прямо или косвенно является подтипом объекта bb. В этом случае атрибут attrib_b2 представлен третьим атрибутом экземпляра объекта xx и ссылается на yy — экземпляр объекта #3:

F: атрибут attrib_x представлен своим значением 4.0.

2 Отображение супертипа, который не является абстрактным (ABSTRACT) супертипом. Определение объекта в языке EXPRESS.

```
ENTITY aa SUPERTYPE OF (ONEOF(bb,dd));  --> A
  attrib_a :  STRING;
END_ENTITY;

ENTITY bb SUBTYPE OF (aa); ----------------------> B
END_ENTITY;
```

```
ENTITY cc SUBTYPE OF (bb); ----------------------> C
  attrib_c :  INTEGER;
END_ENTITY;

ENTITY dd SUBTYPE OF (aa); ----------------------> D
  attrib_d : REAL;
END_ENTITY;

ENTITY ee ; -------------------------------------------------> E
  attrib_e : aa;
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#1 = AA('SAMPLE STRING'); ------------------------> A
#2 = BB('ABC'); ----------------------------------------> B
#3 = CC('DEF', 123);  ----------------------------------> C
#4 = DD('XYZ', 99.99); ---------------------------------> D
#5 = EE(#1); -------------------------------------------> E
#6 = EE(#2); -------------------------------------------> E
#7 = EE(#3); -------------------------------------------> E
#8 = EE(#4); -------------------------------------------> E
```

A: поскольку объект aa не является абстрактным супертипом, то он может присутствовать в структуре обмена. Объект имеет только один атрибут attrib_a, когда присутствует в структуре;

B: объект bb является подтипом aa и, следовательно, его экземпляры будут содержать атрибуты и aa, и bb, но так как объект bb не определяет все атрибуты, в данном атрибуте должен присутствовать только список параметров attrib_a;

C: объект cc является подтипом bb и, следовательно, его экземпляры будут содержать атрибуты aa, bb и cc;

D: объект dd является подтипом aa и, следовательно, его экземпляры должны содержать атрибуты объектов aa и bb;

E: объект ee ссылается на объект aa в качестве атрибута. Следовательно, экземпляр объекта может ссылаться на любой из #1, #2, #3 или #4.

3 Отображение объекта с несколькими супертипами в выражении SUBTYPE OF. Определение объекта в языке EXPRESS.

```
ENTITY base SUPERTYPE OF (branch_one,branch_two); ---> A
  attrib_a :  STRING;
END_ENTITY;

ENTITY branch_one SUBTYPE OF (base); ------------------------> B
  attrib_b :  INTEGER;
END_ENTITY;

ENTITY branch_two SUBTYPE OF (base);  ----------------------> C
  attrib_c :  BOOLEAN;
END_ENTITY;

ENTITY leaf SUBTYPE OF (branch_one, branch_two);  -------> D
  attrib_d : REAL;
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

```
#1 = BASE('SAMPLE STRING');  ----------------------------------> A
#2 = BRANCH_ONE('ABC', 123);  ---------------------------------> B
#3 = BRANCH_TWO('DEF', .T.);  ---------------------------------> C
#4 = LEAF('XYZ', 123, .T., 99.99); ----------------------------> D
```

A: объект не имеет супертипов. При внесении в структуру обмена список его параметров должен содержать только значение атрибута attrib_a;

B: объект branch.one является подтипом основного типа. При внесении в структуру обмена список его параметров должен содержать унаследованные атрибуты, следующие за атрибутом branch_one;

C: объект branch_two является подтипом основного типа. При внесении в структуру обмена список его параметров должен содержать унаследованные атрибуты, следующие за атрибутом branch_two;

D: объект перечисления, являющийся подтипом объектов branch_one и branch_two. При внесении в структуру обмена список его параметров должен содержать унаследованные атрибуты branch_one, включающие в себя атрибуты основного объекта, следующие за унаследованными атрибутами объекта branch_two. Атрибуты основного объекта должны быть описаны однократно при описании атрибутов branch_one. Они должны быть проигнорированы при конфликте с атрибутами объекта branch_two.

12.2.5.3 Внешнее отображение

Если используется внешнее отображение, экземпляр объекта должен быть отображен в COMPLEX_ENTITY_INSTANCE (см. таблицу 3).

ИСО 10303-11 определяет "частное значение сложного объекта" (partial complex entity value) как множество значений атрибутов, описанных единственным EXPRESS-объявлением объекта. Каждое имя типа данных объекта в элементе определяемого множества обозначает частное значение сложного объекта для данного экземпляра объекта. Таким образом, элемент определяемого множества обозначает множество частных значений сложного объекта, что, вместе с именем экземпляра объекта, полностью описывает заданный экземпляр объекта.

Каждое частное значение сложного объекта, обозначаемое именем типа данных объекта в элементе определяемого множества, должно быть отображено в SIMPLE_RECORD (единичная запись) внутри SUBSUPER_RECORD. Порядок SIMPLE_RECORD внутри SUBSUPER.RECORD должен быть возрастающей последовательностью имен типов данных объекта с использованием схемы упорядочения, приведенной в 5.2.

Каждая SIMPLE_RECORD должна кодировать одно частное значение сложного объекта. KEYWORD в каждой SIMPLE_RECORD должно кодировать соответствующее имя типа данных объекта, как определено в 12.2.11, а PARAMETER LIST должен кодировать значения явных атрибутов, если таковые появляются в соответствующем объявлении объекта. Порядок PARAMETER в структуре обмена должен быть тем же, что и порядок соответствующих атрибутов в EXPRESS-объявлении объекта. Если EXPRESS-объявление объекта не содержит явных атрибутов, то PARAMETER_LIST должен быть пустым. Форма каждого PARAMETER должна зависеть от типа данных соответствующего атрибута согласно требованиям 12.1.

Примечания

1 Последовательность частных значений объекта (SIMPLE_RECORD) определяется именем типа данных объекта (так называемым "длинным именем"), а не "сокращенным именем" (если таковое есть), которое может использоваться для кодирования.

2 Каждое частное значение объекта в определяемом множестве должно быть представлено, включая супертипы, не имеющие явных атрибутов.

Примеры

1 Отображение подтипов, связанных посредством ANDOR.

```
ENTITY aa SUPERTYPE OF (bb ANDOR cc);  --> A
  attrib_a :  STRING;
END_ENTITY;

ENTITY bb SUBTYPE OF (aa); ----------------------> B
  attrib_b :  INTEGER;
END_ENTITY;

ENTITY cc SUBTYPE OF (aa);  ---------------------> C
  attrib_c : REAL;
END_ENTITY;

ENTITY dd ;  ------------------------------------------> D
  attrib_d : aa;
END_ENTITY;
Образец экземпляра объекта в секции данных:
#1 = BB('sample string', 15);   ------------------------> A
#2 = CC('S', 3.0);  ----------------------------------------> B
#3 = (AA('ASTRID')BB(17)CC(4.0));  ----------------> C
#4 = DD(#1);  ----------------------------------------> D
#5 = DD(#2);  ----------------------------------------> D
```

#6 = DD(#3); ----------> D
#7 = AA('ABC'); ----------> E

A: #1 является экземпляром комбинации aa и bb;
B: #2 является экземпляром комбинации aa и cc;
C: #3 является экземпляром комбинации aa, bb и cc;
D: объект dd ссылается на объект aa как на атрибут. Следовательно, экземпляр объекта dd может иметь разрешенные ссылки на любой из экземпляров #1. #2 или #3;
E: aa не является абстрактным супертипом и может быть представлен экземплярами, для которых применяется внутреннее отображение, поскольку в этом случае определяемое множество состоит только из одного элемента.

2 Отображение более сложного графа подтип/супертип. Определение объекта в языке EXPRESS.

```
ENTITY x;
  attrib_x : INTEGER;
END_ENTITY;

ENTITY a ABSTRACT SUPERTYPE OF(ONEOF(b,c)); --> A
  attrib_a : x ----------> B
END_ENTITY;

ENTITY b SUPERTYPE OF(d ANDOR e)
    SUBTYPE OF (a);
  attrib_b : REAL; ----------> B
END_ENTITY;

ENTITY c SUBTYPE OF (a); ----------> C
  attrib_c : REAL;
END_ENTITY;

ENTITY d SUBTYPE OF (b); ----------> D
  attrib_d : x;
END_ENTITY;

ENTITY e ABSTRACT SUPERTYPE
      SUBTYPE OF (b); ----------> A
  attrib_e : x; ----------> B
END_ENTITY;

ENTITY f SUPERTYPE OF (h);
  attrib_f : x; ----------> B
END_ENTITY;

ENTITY g SUBTYPE OF (e); ----------> E
  attrib_g : INTEGER;
END_ENTITY;

ENTITY h SUBTYPE OF (e,f); ----------> E
  attrib_h : INTEGER;
END_ENTITY;
```

A: поскольку объекты a и e являются абстрактными супертипами, они не могут присутствовать в структуре обмена в виде самостоятельных экземпляров;
B: поскольку атрибуты attrib_a, attrib_b, attrib_e и attrib_f являются атрибутами объектов супертипа, они будут отображены как унаследованные атрибуты в том случае, если при отображении подтипа применено внутреннее отображение. Эти атрибуты будут отображены как атрибуты объектов, в которых они объявлены, если подтип отображен с использованием внешнего отображения;
C: поскольку объект c участвует в операторе ONEOF, а его супертип не участвует ни в каких операторах с супертипами, для объекта c будет использовано внутреннее отображение;
D: отображение d зависит от структуры определяемого множества, в котором оно появляется;

E: поскольку объекты g и h имеют супертип (объект e), который участвует в операторе ANDOR, их отображение будет зависеть от структуры определяемого множества, в котором они появились.

3 Экземпляр объекта, показывающий внутреннее отображение.

```
#1=X(1);
#2=C(#1, 2.0);
     ^ ^   ^
     | |   |
    A B   C
```

A: определяемым множеством для #2 является [c & a] и, следовательно, используется внутреннее отображение;

B: атрибут attrib_a унаследован типом данных объекта c. Определяемое множество является ссылкой на экземпляр типа данных объекта x;

C: атрибут attrib_c появляется после всех унаследованных атрибутов.

4 Экземпляр объекта, показывающий внутреннее отображение.

```
#4=X(3);
#1=X(1);
#2=D(#1, 2.0, #4)
     ^ ^   ^   ^
     | |   |   |
    A B   C   D
```

A: поскольку экземпляр объекта #2 принадлежит определяемому множеству [a & b & d], которое имеет точно одну вершину (d), он отображается внутренним образом;

B: атрибут объекта a с именем attrib_a унаследован типом данных объекта d;

C: атрибут attrib_b унаследован типом данных объекта d;

D: атрибут attrib_d является последним атрибутом в экземпляре d потому, что атрибуты, унаследованные от объектов супертипа a и b, появляются первыми.

5 Экземпляр объекта, показывающий внешнее отображение.

```
#1=X(1);
#2=(A(#1) B(9.0) D(#1) E(#1) F(#1) H(4) );  ----------> A
```

A: поскольку экземпляр объекта #2 является членом [a & b & d & e & f & h] и это определяемое множество оценки имеет более одной вершины (листа) (d и h), используется внешнее отображение. Не существует единого имени типа данных объекта, которое может быть связано с объектом, скорее объект должен иметь составное имя a-b-d-c-f-h. *Пробелы* между записями объектов необязательны и добавлены в этом примере для того, чтобы облегчить чтение.

12.2.6 Явные атрибуты, переобъявленные как DERIVE

Если объект подтипа переобъявляет атрибут своего супертипа с помощью оператора DERIVE, а исходный атрибут является явным, то значение исходного атрибута в супертипе должно кодироваться *звездочкой* "*".

Пример — Определение объекта в языке EXPRESS.

```
ENTITY point;
  x : REAL;
  y : REAL;
  z : REAL;
END_ENTITY;

ENTITY point_on_curve SUBTYPE OF (point);
  u : REAL;
  c : curve;
DERIVE
  SELF\point.x : real := fx(u, c);
  SELF\point.y : real := fy(u, c);
  SELF\point.z : real := fz(u, c);
END_ENTITY;

ENTITY curve;
```

```
  attr : STRING;
END_ENTITY;
Образец экземпляра объекта в секции данных:
#1 = CURVE('curve_attribute');
#2 = POINT_ON_CURVE( *, *, *, 0.55, #1);  ----------> A
#3 = POINT(2.0, 3.0, 4.0);  ---------------------------------> B
```

A: поскольку здесь используется подтип с вычисляемыми атрибутами, атрибуты x, y и z заменены *звездочками*;

B: поскольку POINT не является абстрактным супертипом, в структуре обмена может присутствовать экземпляр POINT. Атрибуты x, y и z появляются как нормальные атрибуты.

12.2.7 Атрибуты, переобъявленные как INVERSE

Если объект подтипа переобъявляет атрибут своего супертипа с помощью оператора INVERSE, то это не влияет на кодирование. Переобъявленный атрибут в любом случае не кодируют.

12.2.8 Атрибуты, переобъявленные как явные атрибуты

Если объект подтипа переобъявляет атрибут одного из своих супертипов как явный атрибут, т.е. не с помощью оператора INVERSE или DERIVE, то это не влияет на кодирование.

Значение атрибута должно быть закодировано как атрибут супертипа (см. 12.2.5) с применением отображения, определенного в разделе 12 для типа данных атрибута в супертипе. Переобъявленный атрибут должен игнорироваться, т.е. он не должен применяться в качестве атрибута подтипа для целей кодирования.

Пример — Определение объекта в языке EXPRESS.

```
ENTITY aaa SUPERTYPE OF (ONEOF (bbb, ccc));
  a1    : NUMBER;
  a2    : curve;
INVERSE
  a3    : SET OF mmm FOR m1;
END_ENTITY;

ENTITY bbb SUBTYPE OF (aaa);
  SELF\aaa.a1   : INTEGER;
  b             : REAL;
END_ENTITY;

ENTITY ccc SUBTYPE OF (aaa);
  SELF\aaa.a2   : line;
INVERSE
  SELF\aaa.a3   : SET [1:2] OF mmm FOR m1;
END_ENTITY;

ENTITY curve;
  ...
END_ENTITY;

ENTITY line SUBTYPE OF (curve);
  ...
END_ENTITY;

ENTITY mmm;
  m1    : aaa;
END_ENTITY;
Образцы экземпляров в секции данных:
#1 = LINE(...);
#2 = CURVE(...);
#3 = BBB(1.0, #2, 0.5);
#4 = CCC(1.5, #1);
```

Для экземпляров #3 и #4 кодирование такое же, как если бы не было переобъявляемых атрибутов в объектах bbb и ccc.

12.2.9 Локальные правила для объектов

Для объектов локальными являются правила WHERE и UNIQUE, которые не должны отображаться в структуру обмена.

Пример — Определение объекта в языке EXPRESS.

```
ENTITY widget; ----------------------> A
  a : REAL; ---------------------------> B
  b : REAL; ---------------------------> C
  c : REAL; ---------------------------> D
WHERE
  a ** 2 + b ** 2 + c ** 2 = 3.0; -----> E
END_ENTITY;
```

Образец экземпляра объекта в секции данных:

Правила WHERE не представлены в экземпляре объекта.

```
#2 = WIDGET( 1.0, 1.0, 2.0 );
          ^     ^   ^   ^
          |     |   |   |
          A     B   C   D
```

A: имя объекта widget на языке EXPRESS отображено в ключевое слово типа данных объекта для объекта секции данных;

B: в экземпляре объекта атрибут a имеет значение 1.0;

C: в экземпляре объекта атрибут b имеет значение 1.0;

D: в экземпляре объекта атрибут c имеет значение 2.0;

E: правило WHERE не отображено в структуру обмена. Объект синтаксически правилен. Однако значения трех атрибутов не удовлетворяют правилу WHERE.

12.2.10 Отображение инверсных (INVERSE) атрибутов

Атрибуты, описанные в операторе INVERSE, не должны отображаться в структуру обмена.

12.2.11 Кодирование имен типов объектов

Если документ, определяющий схему, экземпляры которой являются предметом секции данных, определяет также и набор сокращенных имен для каждого типа данных объекта в схеме, то эти сокращенные имена должны быть использованы при кодировании имен типов данных объекта. В другом случае именами типов данных объекта при кодировании будут сами имена типов данных объекта. В обоих случаях все строчные буквы должны быть преобразованы в соответствующие прописные буквы, т.е. код не должен содержать никаких строчных букв.

12.3 Отображение элемента EXPRESS для SCHEMA

Элемент EXPRESS для SCHEMA не должен отображаться в структуру обмена. Имя схемы, которая описывает объекты, появляющиеся в структуре обмена, должно быть отображено в заголовочной секции структуры обмена с помощью экземпляра типа данных объекта **file_schema**, как это описано в 8.2.4.

12.4 Отображение элемента EXPRESS для CONSTANT

Каждая ссылка на элемент CONSTANT в EXPRESS должна соответствовать имени экземпляра константы (см. 6.4.4.1) или имени значения константы (см. 6.4.4.2).

12.5 Отображение элемента EXPRESS для RULE

Элементы EXPRESS для RULE не должны отображаться в структуру обмена.

12.6 Комментарии

Комментарии не должны отображаться в структуру обмена.

13 Печатное представление структур обмена

Для управления появлением структуры обмена в печатном виде могут быть использованы комбинации графических символов, как это описано в таблице 6. За исключением указанного ниже, эти

директивы могут быть представлены в любой позиции, где может появиться разделитель лексем (token separator), а также внутри строк (string) и двоичных значений (binaries). Подробности, касающиеся разделителей лексем, приведены в таблице 3. Комбинации графических символов должны появляться вместе без вставки между ними каких-либо графических символов.

Внутри строк также разрешены явные директивы управления процедурой печати. Директивы "\N\" и "\F\" не включают в значимое содержание строки. При кодировании строк см. 6.4.3. Директивы управления процедуры печати "\N\" и "\F\" используют только для печатного представления структуры обмена, и в других случаях они должны игнорироваться. Приложение G представляет руководство по распечатке структуры обмена.

Директивы не должны появляться в лексемах UNIVERSAL_RESOURCE_ IDENTIFIER, в секции привязки (см. раздел 9) или в ссылочной секции (см. раздел 10).

Таблица 6 — Директивы управления печатью

Последовательность графических символов	Значение
\N\ REVERSE_SOLIDUS N REVERSE_SOLIDUS	NEWLINE (новая строка)
F\ REVERSE_SOLIDUS F REVERSE_SOLIDUS	FORMFEED (перевод страницы)

14 Секции подписи

14.1 Структура секции подписи

Синтаксис секции подписи определен в таблице 3. Секция подписи является необязательной. Если секция подписи включена в структуру обмена, то она должна быть расположена после содержания, которое проверяется этой подписью. Может быть несколько секций подписи. Каждая секция должна начинаться со специальной лексемы "SIGNATURE;" и заканчиваться специальной лексемой "ENDSEC;".

Каждая подпись должна проверять содержимое, которое предшествует лексеме "SIGNATURE;", включая любые ранее определенные подписи.

Подпись должна быть структурирована в соответствии с синтаксисом криптографических сообщений (CMS) для подписи с внешним контентом. Определение CMS приведено в 3.1.7.6.

Примечание — Внешним контентом, поскольку данные включены в разделы над подписью и не включены в подпись.

При вычислении **message_digest** для CMS реализация должна включать только символы алфавита, определенного в таблице 1. Определение **message_digest** приведено в 3.1.7.7.

Структура CMS должна быть записана в структуру обмена с использованием кодирования base64. Определение base64 см. в 3.1.7.5.

Пример — Три строки подписи.

```
SIGNATURE;
MIIGpgYJKoZIhvcNAQcCoIIGlzCCBpMCAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3
DQEHAaCCA9cwggPTMIICu6ADAgECAgEEMA0GCSqGSIb3DQEBCwUAMHoxEzARBgoJ
kiaJk/IsZAEZFgNjb20xGTAXBgoJkiaJk/IsZAEZFglzdGVwdG9vbHMxFzAVBgNV
...
ENDSEC;
```

Примечания

1 Содержание подписи описано в RFC 5652, раздел 5. Кодирование подписи описано в RFC 4648, раздел 4.

2 Согласно таблице 3, первая подпись дается после лексемы "ISO-10303-21;", а остальные следуют за каждым подписываемым содержанием, проверяя все данные алфавита, которые предшествуют символу "S" в лексеме SIGNATURE.

Приложение А
(обязательное)

Представление файла на носителе данных

А.1 Перенос контента с доступом к записям

Магнитная лента может содержать несколько наборов данных. Каждый набор данных может быть последовательным файлом, содержимое которого может быть интерпретировано как структура обмена, соответствующая настоящему стандарту.

Отправитель такой ленты несет ответственность за передачу любому получателю либо на ленте, либо иным образом информации о том, какие наборы данных являются структурами обмена, соответствующими настоящему стандарту.

Формат передачи для каждой структуры обмена выглядит следующим образом:

- файл должен состоять из последовательности записей;
- первые графические символы в первой записи должны быть специальной лексемой "ISO-10303-21;", инициирующей структуру обмена. Форматирование места хранения носителя зависит от операционной системы и является предметом согласования между отправителем и получателем;
- последняя запись должна быть заполнена (при необходимости) *пробелами после точки с запятой* специальной лексемы "END-ISO-10303-21;".

А.1.1 Формат передачи для носителя на магнитной ленте

Для передачи на ленте обменных файлов предпочтительный формат обмена определяют следующие характеристики, установленные в ИСО/МЭК 3788:

- ширина ленты 12,7 мм (0,5 дюйма):
- неразмеченная лента;
- девятидорожечная лента;
- плотность записи на ленте — 63 символ/мм (1600 бит/дюйм);
- размер физического блока — 4000 октетов (8-битных байтов);
- блоки разделены пропусками между записями;
- за последним блоком должен следовать маркер ленты.

Также может быть использован другой стандарт носителя на магнитной ленте, например магнитная лента с плотностью записи 246 символ/мм (6250 бит/дюйм).

А.1.2 Другие носители данных с доступом к записи

Другие носители, на которых структуры обмена хранятся как последовательности записей, могут использовать также формат передачи, определенный для лент.

А.2 Перенос контента с доступом к строкам

Некоторые магнитные носители содержат наборы данных, хранящиеся как последовательность строк. Каждый из этих наборов может быть последовательным файлом, содержание которого может быть интерпретировано как файл, соответствующий настоящему стандарту.

Файл должен состоять из последовательности строк:

- первые графические символы в первой строке должны быть специальной лексемой "ISO-10303-21:", которая инициирует структуру обмена;
- последняя строка должна быть дополнена при необходимости пробелами за специальной лексемой "END-ISO-10303-21;", завершающей структуру обмена.

Средства ограничения строк и представления конца файла зависят от операционной системы и являются предметом соглашения между отправителем и получателем с учетом того, что ограничители не должны использовать каких-либо символов основного алфавита. При любом их представлении ограничители строк и файла должны игнорироваться при обработке структуры обмена.

А.2.1 Формат передачи для носителя на дискете

Для передачи структуры обмена может быть использован любой из популярных носителей для дискет (3 1/2″, 5 1/4″, низкая или высокая плотность).

Отправитель такой дискеты несет ответственность за передачу любому получателю либо на дискете, либо за ее пределами информации о том, какие наборы данных являются структурами обмена, соответствующими настоящему стандарту.

А.2.2 Другие носители

Другие носители, на которых файлы хранятся как последовательности строк, могут использовать тот же формат передачи, который определен для дискет. В частности, этот формат может быть пригоден для передачи через сети связи (E-mail).

Примечания

1 Во время передачи файлов с открытым текстом по электронной почте строки, начинающиеся с *точки*, иногда искажаются, либо точка пропадает, либо она удваивается. При передаче по электронной почте может происходить перенос строки во вложениях с открытым текстом, поэтому также наблюдается потеря или удвоение точки, появляющейся позже в длинных строках.

2 В реализациях рекомендуется использовать разрывы строк в структуре обмена, чтобы избежать размещения точки в качестве первого символа и переноса строки, содержащей *точку*, чтобы длина строки была не более 80 символов. Размещение структуры обмена в ZIP-архиве, как описано в А.4, также позволяет избежать проблемы, поскольку файлы ZIP обычно при отправке в виде вложений электронной почты обрабатываются по-другому.

А.3 Обработка многотомных файлов

Может быть необходимым распределить структуру обмена на нескольких физических томах.

Примечание — В зависимости от конкретных условий объединять физически раздельные тома многотомного файла в одну многотомную структуру можно с помощью специальных программных средств или операционной системы.

А.4 Передача содержимого сжатого архива

Файл, соответствующий требованиям данной спецификации, может быть сжат и сохранен в архиве ZIP (см. 3.1.6). Архив должен быть написан с использованием сжатия PKZip 2.04g.

Примечание — Этот метод сжатия ограничивает поддержку zip, чтобы исключить шифрование, поддержку имени файла Юникода и использование механизма deflate64. Это сжатие совместимо с Windows Compressed Folders, WinZip, info-zip и zlib.

Архив ZIP может содержать несколько сжатых файлов. Некоторые из этих файлов могут представлять собой структуры обмена, удовлетворяющие требованиям данной спецификации. Некоторые из файлов могут быть поддиректориями, содержащими дополнительные файлы. Некоторые файлы могут сами быть ZIP-архивами. Наконец, некоторые из файлов могут быть вспомогательными данными, которые не являются частью информационной модели, но переносятся с моделью, чтобы на них можно было ссылаться.

Если в сжатом архиве распознается URI, то корневой файл в архиве считывается в приложение. Этот файл должен представлять собой структуру обмена, отвечающую ограничениям настоящего стандарта. Этот файл должен иметь имя "ISO-10303.p21".

Примечание — Транспортный протокол (например, http, ftp, file и т. д.) обеспечивает доставку файла запрашивающей стороне. Если этот файл является ASCII, то он обрабатывается как структура обмена. Если файл является двоичным, то применяется алгоритм декомпрессии и файл с именем "ISO-10303.p21" обрабатывается как структура обмена.

Файл "ISO-10303.p21" должен называться корневым. Любые другие файлы в архиве, включая любые файлы в поддиректориях, должны быть дополнительными. Архив должен соответствовать настоящему стандарту, только если он содержит корень.

У корневого файла должно быть свое локальное имя, если на него ссылаются другие файлы внутри архива, и имя архива, когда на него ссылаются извне. Для обеспечения согласованности ссылок в архиве все относительные адреса должны интерпретироваться как адреса относительно местоположения ссылочного файла в архивном каталоге. Ни один относительный адрес не должен адресовать файл вне архива.

Примечание — Поскольку в области применения вся относительная адресация поддерживается, то реализация может считывать файл ZIP и распаковывать его в выбранном месте для последующей обработки.

За пределами архива можно ссылаться только на корневой файл. Если другие файлы в архиве содержат объекты, на которые требуется внешняя ссылка, то в корневом файле должна быть определена привязка, и эта привязка должна переслать ссылку на требуемый объект или значение в дополнительном файле.

Примеры

1 Для здания создается архив. Корневой файл описывает здание. Дополнительные файлы описывают два этажа и картину.

Имя файла	Краткое описание содержимого.
ISO-10303.p21	Данные заголовка проекта.
first_floor.ifc	Описание первого этажа.
second_floor.ifc	Описание второго этажа.
picture.jpeg	Фотография здания.

2 Секция привязки файла ISO-10303.p21 предоставляет архивные привязки внешним приложениям.

ANCHOR;
<first_floor> = #10 /* первый этаж, определение которого содержится в другом файле (см. ссылки) */
<second_floor> = #20 /* второй этаж, определение которого содержится в другом файле (см. ссылки) */
<electrical_system> = #30 /* электрическая система, определение которой содержится в том же файле */
<building_photo> = <picture.jpeg> /* дополнительные данные, не являющиеся частью информационной модели — но делающие доступной*/
ENDSEC;

3 Ссылочная секция файла ISO-10303.p21.

REFERENCE;
#10 = <first.ifc#51c13346-2084-4494-8003-d956d1d25c45> /* GUID, привязывающий к экземпляру объекта, находящемуся в файле first.ifc */
#20 = <second.ifc#cdd73fcb-e2d1-4546-b4f2-c34d2d333d42> /* GUID, привязывающий к экземпляру объекта, находящемуся в файле second.ifc */
ENDSEC;

А.5 Передача содержимого директории

Для того чтобы адресация была согласованной для структур, которые могут находиться в определенное время в архиве, а в другое время — вне архива, также должна применяться передача правил сжатого содержимого, если URI разрешается в директории, содержащий файл с именем "ISO-10303.p21".

Приложение В
(обязательное)

Соглашения по записи в синтаксической нотации Вирта

Синтаксис структуры обмена определен в синтаксической нотации Вирта (WSN), опубликованной Никлаусом Виртом в "Communications of the ACM. 20:11 (Nov 77). 822-823". WSN состоит из набора выводов или правил подстановки. Элемент в левой части вывода (т. е. перед знаком равенства) может быть использован для представления появления образца в правой части. Элементарные символы, которые появляются только в правой части таких выводов, называются терминальными. Элементы, которые появляются в левой части выводов, называются нетерминальными.

Соглашения по записи даны ниже. Таблица В.1 представляет самоопределенную WSN:

- строка *прописных букв* является элементом языка: строка является именем элемента (для удобства *строчные буквы* используются для неопределенных идентификаторов и комментариев);
- любая строка, заключенная в *кавычки*, точно определяет содержимое внутри *кавычек*. Единственным исключением из этого правила является *знак кавычек* внутри текста. Чтобы его реализовать, *знак кавычек* сразу же повторяют один раз. Последовательность """" интерпретируется как ", а последовательность "AB""C" — как AB"C;
- *знак равенства* "=" означает вывод. Определено, что элемент слева должен быть комбинацией элементов справа. Любые *пробелы*, появляющиеся между элементами вывода, не являются значащими, если они появляются не внутри литерала. Вывод завершается *точкой*;
- *фигурные скобки* "{ }" означают отсутствие или множество повторений;
- *квадратные скобки* "[ ]" означают необязательный параметр;
- *вертикальная линия* "|" означает логическое ИЛИ;
- *скобки* "(" ")" показывают приоритет операций. В частности там, где скобки включают в себя элементы, разделенные *вертикальными линиями*, один из элементов должен быть выбран в сочетании с любой другой операцией.

Пример — Последовательность "A(B|C|D)" эквивалентна "AB|AC|AD".

Таблица В.1 — Самоопределенная синтаксическая нотация Вирта (WSN)

```
SYNTAX        = { PRODUCTION }.

PRODUCTION    = IDENTIFIER "=" EXPRESSION ".".

EXPRESSION    = TERM { "|" TERM }.

TERM          = FACTOR { FACTOR }.

FACTOR        = IDENTIFIER
                    | LITERAL
                    | "[" EXPRESSION "]"
                    | "(" EXPRESSION ")"
                    | "{" EXPRESSION "}" .

IDENTIFIER    = letter { letter }.

LITERAL       = """" character { character } """".
```

Приложение С
(обязательное)

Регистрация информационного объекта

С.1 Обозначение документа

Для однозначного обозначения информационного объекта в открытой системе настоящему стандарту присвоен следующий идентификатор объекта:

{ ISO standard 10303 part(21) version(4) }

Смысл данного обозначения установлен в ИСО/МЭК 8824-1 и описан в ИСО 10303-1.

С.2 Обозначение схемы

Для однозначного обозначения в открытой информационной системе схеме **header_section_schema**, установленной в настоящем стандарте (см. раздел 8), присвоен следующий идентификатор объекта:

{ ISO standard 10303 part(21) version(5) object(1) header-section-schema(1) }

Смысл данного обозначения установлен в ИСО/МЭК 8824-1 и описан в ИСО 10303-1.

Приложение D
(обязательное)

Форма заявки о соответствии реализации протоколу

D.1 Заявление о соответствии

Форма заявки о соответствии реализации протоколу (ЗСРП) предназначена для обеспечения оценки соответствия реализаций настоящему стандарту. Настоящее приложение составлено в форме вопросника. Данный вопросник должен быть заполнен реализатором и может быть использован испытательной лабораторией при подготовке аттестационного тестирования заявленной реализации.

Все исполнители должны предоставить ответы на поставленные вопросы.

D.2 Класс соответствия

Контрольная проверка 1.

Обеспечивает ли реализация класс синтаксического соответствия 1:

___ для чтения ___ для записи?

Обеспечивает ли реализация класс синтаксического соответствия 2:

___ для чтения ___ для записи?

Обеспечивает ли реализация класс синтаксического соответствия 3:

___ для чтения ___ для записи?

D.3 Кодирование

Выполнить необходимое количество контрольных проверок.

D.3.1 Кодирование экземпляра объекта

Обеспечивает ли реализация представление имен экземпляров объектов:

___ для чтения ___ для записи?

Обеспечивает ли реализация представление имен экземпляров значений:

___ для чтения ___ для записи?

Обеспечивает ли реализация представление имен констант EXPRESS:

___ для чтения ___ для записи?

D.3.2 Кодирование сокращенных имен

Обеспечивает ли реализация представление сокращенных имен объектов:

___ для чтения ___ для записи?

Обеспечивает ли реализация представление сокращенных имен для значений выбранных типов:

___ для чтения ___ для записи?

Обеспечивает ли реализация представление сокращенных имен для перечисляемых значений:

___ для чтения ___ для записи?

D.3.3 Кодирование строки

Обеспечивает ли реализация кодирование в формате \X\ для символов:

___ для чтения: если да, то каково двоичное представление, использованное в реализации?

___ для записи?

Обеспечивает ли реализация кодирование символов в форматах \S\ и \P\ по стандартам серии ИСО/МЭК 8859:

___ для чтения: если да, то каково двоичное представление, использованное в реализации?

___ для записи?

Обеспечивает ли реализация кодирование символов в форматах \X2\ по стандартам серии ИСО/МЭК 10646:

___ для чтения; если да, то каково двоичное представление, использованное в реализации?

___ для записи?

Обеспечивает ли реализация кодирование символов в форматах \X4\ по стандартам серии ИСО/МЭК 10646:

___ для чтения; если да, то каково двоичное представление, использованное в реализации?

___ для записи?

Обеспечивает ли реализация кодировку UTF-8 для символов ИСО/МЭК 10646 для структур обмена, заявляющих значение **implementation_level** "4;1" или "4;2", или "4;3" (см. 8.2.2):

___ для чтения; если да, то каково двоичное представление, использованное в реализации?

___ для записи?

D.3.4 Кодировка ссылок

Поддерживает ли реализация ссылки URI:

___ с привязками ___ со ссылками ___ в открытых текстах ___ в сжатых архивах?

Управляет ли реализация допустимостью ссылок с помощью явного заполнения схемы:

___ без временных меток ___ с временными метками ___ без подписей ___ с подписями?

D.4 Ограничения реализации

Каково максимальное число схем, на которые можно ссылаться в структуре обмена?

Каково максимальное число секций данных, входящих в структуру обмена?

Каково максимальное количество имен экземпляров (или двоичное представление, использованное в реализации), входящих в структуру обмена?

Каковы максимальные и минимальные значения (или двоичное представление, использованное в реализации) для целых чисел (INTEGER) в соответствии с языком EXPRESS?

Каковы ограничения по точности (или двоичное представление, использованное в реализации) для вещественных чисел (REAL) в соответствии с языком EXPRESS?

Какова максимальная длина строки (STRING) в соответствии с языком EXPRESS?

Какова максимальная длина двоичного числа (BINARY) в соответствии с языком EXPRESS?

Каково максимальное число элементов, могущих входить в агрегат (массив) при его кодировании?

Какова максимальная глубина вложения при кодировании вложенных агрегатов (массивов)?

Приложение Е
(обязательное)

Множество EXPRESS-схем в структуре обмена

Е.1 Допустимые ссылки

В структуре обмена с несколькими секциями данных могут быть приведены ссылки на экземпляры объектов, определенные в секциях данных по различным EXPRESS-схемам. При реализации ссылок между схемами возникают два вопроса: 1) какие ссылки можно считать верными? и 2) какие экземпляры следует считать правильными, основанными на конкретной схеме, при определении схематического соответствия структуре обмена?

В настоящем разделе описаны два метода определения правильности ссылок между экземплярами объектов при установлении схематического соответствия для структуры обмена. Можно использовать другие методы, не определенные в настоящем стандарте.

Примечание — В данном разделе описаны способы, посредством которых автор документов, определяющих EXPRESS-схемы, также может задать правильные ссылки между двумя или несколькими схемами.

При определении схематического соответствия реализация может ссылаться на конкретный документ, описывающий данную схему в рамках определений на языке EXPRESS, сокращенных имен и других требований или ограничений. Если заданную схему используют в сочетании с другими, данный документ так же должен определять правильность ссылок между этими схемами.

Если предполагается, что данное определение делается однократно и охватывает ряд схем, то не следует дополнительно определять верность ссылок в конкретной структуре обмена.

Е.1.1 Метод спецификации интерфейса на языке EXPRESS

При использовании данного метода ссылки между экземплярами объектов различных схем должны быть заданы посредством определения интерфейса на языке EXPRESS в соответствии с разделом 11 ИСО 10303-11.

На экземпляр типа, заданный в схеме, можно ссылаться через экземпляр типа, определенный в другой схеме, если этот тип связан с последней посредством конструктивов USE или REFERENCE.

Пример — Рассматриваются две схемы и основанная на них структура обмена.

```
SCHEMA base;
ENTITY a;
  range : REAL;
END_ENTITY;
ENTITY b;
  name : STRING;
END_ENTITY;
END_SCHEMA;

SCHEMA extension;
USE FROM base (a, b);

ENTITY c;
  addressed_item : b;
  address : STRING;
END_ENTITY;

RULE a_range_positive FOR (a);
WHERE
  WR1: SIZEOF (QUERY (inst <* a | inst.range < 0)) = 0;
END_RULE;
END_SCHEMA;


ISO-10303-21;
HEADER;
..
..
```

```
FILE_SCHEMA(('BASE', 'EXTENSION'));
ENDSEC;
DATA ('ONE', ('BASE'));
#1=A(-3.5);
#2=B('Sam Smith');
#3=B('John Doe');
ENDSEC;
DATA ('TWO', ('EXTENSION'));
#4=C(#2, '100 Main Street');
#5=C(#3, '1300 Elmwood Avenue);
ENDSEC;
END-ISO-10303-21;
```

При использовании метода задания требований к интерфейсу на языке EXPRESS для определения правильности ссылок реализация должна учитывать, что объект типа B явно связан со схемой EXTENSION, поэтому допустимы ссылки из #4 на #2 и из #5 на #3.

E.1.2 Метод определения эквивалентности области значений SDAI

При использовании данного метода ссылки между экземплярами объектов, заданными в разных схемах, должны быть выполнены на основе методов определения эквивалентности области значений, установленных в ИСО 10303-22.

При задании экземпляра типа, определенного в схеме, экземпляры любых типов объектов из данной схемы, ссылающиеся на данный тип. также могут ссылаться на экземпляры любых типов, объявленных в эквивалентной области значений первого типа.

Пример — Рассматриваются две схемы и основанная на них структура обмена.

```
SCHEMA LONGA;
ENTITY a;
  range : REAL;
END_ENTITY;
ENTITY b;
  name : STRING;
END_ENTITY;
END_SCHEMA;

SCHEMA LONGB;
ENTITY a;
  range : REAL;
END_ENTITY;
ENTITY b;
  name : STRING;
END_ENTITY;

ENTITY c;
  addressed_item : b;
  address : STRING;
END_ENTITY;

RULE a_range_positive FOR (a);
WHERE
  WR1: SIZEOF (QUERY (inst <* a | inst.range < 0)) = 0;
END_RULE;
END_SCHEMA;


ISO-10303-21;
HEADER;
..
..
FILE_SCHEMA(('LONGA', 'LONGB'));
ENDSEC;
```

```
DATA ('ONE', ('LONGA'));
#1=A(-3.5);
#2=B('Sam Smith');
#3=B('John Doe');
ENDSEC;
DATA ('TWO', ('LONGB'));
#4=C(#2, '100 Main Street');
#5=C(#3, '1300 Elmwood Avenue);
ENDSEC;
END-ISO-10303-21;
```

В настоящем примере предполагается наличие информации об области эквивалентности значений, заданной с использованием одного из методов, описанных в ИСО 10303-22, устанавливающей эквивалентность области значений типов A и B в обеих схемах. Это представлено в виде:

LONGA.A is DEQ to LONGB.A
LONGB.A is DEQ to LONGB.A
LONGA.B is DEQ to LONGB.B
LONGB.B is DEQ to LONGA.B

При использовании области эквивалентности для определения правильности ссылок реализация должна учитывать, что тип объекта LONGA.B является областью эквивалентности значений для LONGB.B, что допускает ссылки из #4 на #2 и из #5 на #3.

E.2 Определение совокупности схемы

Объект **file_population** в заголовочной секции связывает EXPRESS-схему и набор (коллекцию) экземпляров объектов с конкретной структурой обмена. Атрибут **determination_method** определяет алгоритм выбора коллекции экземпляров объектов, заданных в наборе секций данных. В настоящем разделе описаны три метода их определения. Можно использовать другие методы, не определенные в настоящем стандарте.

При определении схематического соответствия структуры обмена коллекция экземпляров объектов, заданная в соответствующих **file_population**, должна быть проверена на соответствие EXPRESS-схеме. Если на какую-либо секцию данных нет ссылки из какого-либо объекта **file_population**, то эта секция должна быть проверена на соответствие определенной в ней схемы по методу ограничения секции, описанному ниже.

При проверке соответствия схемы установленным требованиям и ограничениям ссылки на экземпляры объектов, сделанные вне коллекции данных экземпляров, должны восприниматься как ошибочные.

Примечание — Последующая процедура описывает метод проверки структуры обмена в соответствии с 8.2.4 и вышеизложенным параграфом.

Для каждого объекта **file_population**, имеющего в структуре обмена значение "F":

- находят набор экземпляров посредством использования метода, заданного объектом **F.determination_method**, для секций данных, поименованных в объекте **F.governed_sections**. Если объект **F.governed_sections** является пустым (неопределенным), то данный метод используют для всех секций данных в структуре обмена;
- проверяют данный набор на соответствие правилам и ограничениям, заданным в объекте **F.governing_schema**;
- отмечают секции данных, являющиеся исходными данными для объекта **F.determination_method**.

Для каждой немаркированной секции данных "D" проверяют набор экземпляров в этой секции на соответствие правилам и ограничениям, заданным в схеме.

E.2.1 Метод ограничения секции

При использовании метода ограничения секции атрибут **determination_method** должен иметь значение "SECTION_BOUNDARY". Набор (коллекция) экземпляров объектов, заданный в качестве исходных данных для одной или нескольких секций, должен содержать:

- все экземпляры из заданной секции данных.

Примеры

1 Рассмотрим схемы и структуру обмена, описанные в примере согласно F.1.1. Заголовочная секция не содержит каких-либо экземпляров объекта file_population. При определении схематического соответствия структуры обмена должно быть учтено следующее:

- на секцию данных ONE не ссылаются из какого-либо объекта file_population, поэтому следует проверить все экземпляры объектов этой секции на соответствие задающей ее схеме. Все экземпляры в секции данных ONE должны удовлетворять требованиям и ограничениям схемы BASE. В данном примере совокупность удовлетворяет всем ограничениям схемы BASE;

- на секцию данных TWO не ссылаются из какого-либо объекта fde_population, поэтому следует проверить все экземпляры объектов этой секции на соответствие задающей ее схеме. Все экземпляры в секции данных ONE должны удовлетворять требованиям и ограничениям схемы EXTENSION.

В настоящем примере секция данных не содержит каких-либо экземпляров объекта A, поэтому она удовлетворяет правилу a_range_positive. Атрибут addressed_item экземпляров #4 и #5 ссылается на экземпляры вне данной совокупности, поэтому эти ссылки следует рассматривать как ошибочные. Атрибут addressed_item не является факультативным (необязательным). Поэтому данная совокупность не удовлетворяет ограничениям схемы EXTENSION.

2 Рассмотрим схемы и структуру обмена, описанные в примере согласно E.1.1, но с заголовочной секцией, приведенные ниже.

```
HEADER;
..
..
FILE_SCHEMA(('BASE', 'EXTENSION'));
file_population('BASE', 'SECTION_BOUNDARY', ('ONE'));
file_population('EXTENSION', 'SECTION_BOUNDARY', ('ONE','TWO'));
ENDSEC;
```

При определении схематического соответствия структуры обмена должно быть учтено следующее:

- первый объект file_population определяет набор (коллекцию) экземпляров, управляемых схемой BASE. Все экземпляры в секции данных ONE должны удовлетворять требованиям и ограничениям схемы BASE. В данном примере совокупность удовлетворяет всем ограничениям схемы BASE;

- второй объект file_population определяет набор (коллекцию) экземпляров, управляемых схемой EXTENSION. Все экземпляры в секции данных ONE и TWO должны удовлетворять требованиям и ограничениям схемы EXTENSION. В рассматриваемом примере правило a_range_positive нарушено в экземпляре #1. Поэтому данная совокупность не соответствует ограничениям схемы EXTENSION.

E.2.2 Описание всех совместимых методов

Атрибут **determination_method** должен иметь значение 'INCLUDE-ALL-COMPATIBLE' в случае использования всех совместимых методов. Набор (коллекция) экземпляров объектов, заданный в качестве исходных данных для одной или нескольких секций данных, должен охватывать:

- все экземпляры в регулируемых секциях данных;
- все экземпляры в других секциях данных, где тип объекта экземпляра позволяет ссылаться на экземпляры, заданные схемой управления совокупностью.

Пример — Рассмотрим схемы и структуру обмена, описанные в примере согласно E.1.1, но с заголовочной секцией, приведенные ниже.

```
HEADER;
..
..
FILE_SCHEMA(('BASE', 'EXTENSION'));
file_population('BASE', 'INCLUDE_ALL_COMPATIBLE', ('ONE'));
FILE_POPULATION('EXTENSION', 'INCLUDE_ALL_COMPATIBLE', ('TWO'));
ENDSEC;
```

При определении схематического соответствия структуры обмена должно быть учтено следующее:

- первый объект **file_population** определяет набор (коллекцию) экземпляров, управляемых схемой BASE. Данный набор содержит все экземпляры из секции данных ONE. Он так же должен содержать экземпляры из секции данных TWO, допускающие ссылки на экземпляры, заданные схемой BASE. В настоящем примере это не рассмотрено. Однако секция данных TWO включает экземпляры типов A и B, которые могут быть ограниченными. В данном примере совокупность удовлетворяет всем ограничениям схемы BASE:

- второй объект **file_population** определяет набор (коллекцию) экземпляров, управляемых схемой EXTENSION. Этот набор содержит все экземпляры из секции данных TWO. Он также содержит все экземпляры типов A и B из секции данных ONE, потому что их типы позволяют ссылаться на экземпляры, определенные схемой EXTENSION. В рассматриваемом примере правило **a_range_positive** нарушено в экземпляре #1, поэтому данная совокупность не соответствует ограничениям схемы EXTENSION.

E.2.3 Описание метода ссылки на экземпляр

Атрибут **determination_method** должен иметь значение 'INCLUDE_REFERENCED' в случае использования метода ссылки на экземпляр. Набор (коллекция) экземпляров объектов, заданный в качестве исходных данных для одной или нескольких секций данных, должен охватывать:

- все экземпляры в регулируемых секциях данных;
- экземпляры из других секций данных, которые ссылаются на экземпляры в заданных секциях данных.

Пример — Рассмотрим схемы и структуру обмена, описанные в примере согласно F.1.1, но с заголовочной секцией, приведенные ниже:

```
HEADER;
..
..
FILE_SCHEMA(('BASE', 'EXTENSION'));
FILE_POPULATION('BASE', 'INCLUDE_REFERENCED', ('ONE'));
FILE_POPULATION('EXTENSION', 'INCLUDE_REFERENCED', ('TWO'));
ENDSEC;
```

При определении схематического соответствия структуры обмена должно быть учтено следующее:

- первый объект file_population определяет набор (коллекцию) экземпляров, управляемых схемой BASE. Данный набор содержит все экземпляры из секции данных ONE. Он также должен содержать экземпляры из секции данных TWO, допускающие ссылки на экземпляры, заданные в секции данных ONE. В настоящем примере это не рассмотрено. В данном примере совокупность удовлетворяет всем ограничениям схемы BASE;

- второй объект file_population определяет набор (коллекцию) экземпляров, управляемых схемой EXTENSION. Этот набор содержит все экземпляры из секции данных TWO. Он также содержит все экземпляры #2 и #3 из секции данных ONE, потому что их типы позволяют ссылаться на экземпляры из секции данных TWO. В рассматриваемом примере экземпляр #1 не является членом совокупности, поэтому правило a_range_positive удовлетворено. Данная совокупность соответствует всем ограничениям схемы EXTENSION.

Приложение F
(обязательное)

Привязка ECMAScript к секции привязки

F.1 Введение

Описанная в настоящем приложении привязка отображает секцию привязки обменной структуры в объекты ECMAScript.

Примечания

1 Термин ECMAScript используется в ИСО/МЭК 16262 для языка, более известного как JavaScript.

2 Элементы привязки могут ссылаться на экземпляры данных в теле структуры обмена.

Привязка позволяет материализовать данные, на которые ссылаются элементы привязки, как объекты, которые могут обрабатываться приложением.

Привязка определяется как набор функций, выполняемых в контексте, определяемом объектом, называемым объект P21.

F.2 Требуемые свойства объекта P21

Объект P21, доставляемый функциями, которые читают обменную структуру, должен иметь одно свойство на каждый элемент привязки в обменной структуре. Это свойство должно иметь то же имя, что и элемент привязки.

Пример — В следующем коде синхронная функция чтения используется для чтения обменной структуры и доступа к элементу привязки под названием 'geometry' (геометрия).

```
function read_model_geometry()
{
   var model = read_model(“example.p21”);
   return model.geometry;
}
```

Примечание — Функция **read_model** является лишь примером. Для достижения лучшей производительности могут использоваться функциональные средства асинхронного чтения.

Когда приложение читает обменную структуру, оно по значению атрибута **file_name.name** (наименование) из заголовочной секции должно определить, какие элементы привязки ожидаются. Если две обменные структуры имеют одинаковые значения атрибута **file_name.name** (наименование), то каждый элемент привязки с тем же наименованием будет иметь то же представление.

Примечание — Например, если атрибут **file_name.name** (наименование) имеет значение 'workingstep. paths', то можно ожидать, что в обменной структуре имеются элементы привязки, представляющие траектории инструмента.

F.3 Отображение значений из секции привязки

Каждый элемент привязки должен иметь свойства со следующими описаниями:

свойство, называемое "$value", которое описывает значение элемента привязки, кодированное как описано ниже;

каждая метка элемента привязки будет дополнительным свойством, с именем метки, которое начинается со *знака доллара* ("$") и кодируется так, как описано ниже.

Примеры

1 Выражение ECMAScript "model.first.$value = new P21.Integer (10);" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
<first> = 10;
ENDSEC;
```

2 Два выражения ECMAScript "model.second.$value = new P21.Real (10);" и "model.second.$third = new P21.String ("10");" эквивалентны следующему коду в структуре обмена:

```
ANCHOR;
<second> = 10. {third:'10'};
ENDSEC;
```

Каждое значение элемента привязки должно быть представлено как объект P21.Wrapper. Каждый экземпляр должен быть одним из перечисленных ниже подтипов P21.Wrapper. Объект P21.Wrapper должен иметь метод под названием toP21String(), в дополнение к методам toString() и valueOf(), которые предписаны в ECMAScript.

Если экземпляр не имеет подтипов, метод toP21String() должен вернуть строку со значением '$'.

Примечание — Значение, лежащее в основе свойства wrapper, доступно, когда это необходимо, через метод valueOf(), а используемое для печати представление доступно через метод toString().

F.3.1 Отображение целых значений

Свойство, определенное ключевым словом INTEGER (целое) отображается в объект P21.Integer. Метод valueOf() возвращает целочисленный литерал, представленный как число ECMAScript. Метод toString() возвращает представление числа ECMAScript в виде строки ECMAScript. Метод toP21String() возвращает представление числа ECMAScript в виде строки, соответствующей требованиям к целочисленным литералам, которые определены в настоящем стандарте.

Примечание — Во многих примерах в настоящем приложении предполагается, что объект P21 читает и присваивает переменную, называемую 'model' (модель).

Пример — Выражение ECMAScript "model.example.$value = new P21.Integer (10);" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = 10;
..
ENDSEC;
```

Примечание — В ECMAScript нет целого типа. Все числа относятся к численному типу.

F.3.2 Отображение действительных значений

Свойство, определенное ключевым словом REAL (действительное), отображается в объект P21.Real. Метод valueOf() возвращает литерал действительного типа, представленный как число ECMAScript. Метод toString() возвращает представление числа ECMAScript в виде строки ECMAScript. Метод toP21String() возвращает представление числа ECMAScript в виде строки, соответствующей требованиям к литералам действительного типа, которые определены в настоящем стандарте.

Пример — Выражение ECMAScript "model.example.$value = new P21.Real (10);" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = 10.;
..
ENDSEC;
```

Примечание — В настоящем стандарте требуется, чтобы представление действительных чисел включало десятичную точку ".".

F.3.3 Отображение строк

Свойство, определенное ключевым словом STRING (строка) отображается в объект P21.String. Метод valueOf() возвращает строчный литерал без начального и завершающего символов " ' ". Метод toString() возвращает точно такой же результат, что и метод valueOf(). Метод toP21String() возвращает строку, соответствующую требованиям к текстовым литералам, которые определены в настоящем стандарте. Строка символов должна начинаться и заканчиваться символами " ' ".

Пример — Выражение ECMAScript "model.example.$value = new P21.String ('This is a message');" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = 'This is a message';
..
ENDSEC;
```

F.3.4 Отображение перечислимых типов

Свойство, определенное ключевым словом ENUMERATION (перечислимое) отображается в объект P21. Enumeration. Метод valueOf() возвращает значение ECMAScript true (истина), если в структуре обмена присутствовало значение ".T.". Если в структуре обмена присутствовало значение ".F.", то метод возвращает значение ECMAScript false (ложь). Иначе, если в структуре обмена присутствовало значение перечислимого типа, метод возвращает символьную строку ECMAScript без символов "." в первой и последней позициях. Метод toString() возвращает результат ECMAScript применения метода toString() к результату метода valueOf(). Метод toP21String() возвращает символьную строку со значением перечислимого типа в формате, определение формата содержится в настоящем стандарте.

Примеры

1 Выражение The ECMAScript "model.example.$value = new P21.Enumeration ('RED');" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = .RED.;
..
ENDSEC;
```

2 Выражение ECMAScript "model.example.$value = new P21.Enumeration (true);" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = .T.;
..
ENDSEC;
```

Примечания

1 Значения 'истина' и 'ложь' переменных языка EXPRESS типа BOOLEAN (булева переменная) и LOGICAL (логическая переменная) представляются как ".T." и ".F.", соответственно. В принципе, другие переменные перечислимого также могут иметь те же значения. В этом случае отображение может привести к неожиданным результатам, но никакой потери информации не произойдет.

2 В среде ECMAScript пустые значения неявно преобразуются в булевы значения 'ложь'. Это вызовет затруднения для значений OPTIONAL BOOLEAN (необязательное значение булева типа) в реализации SDAI, но для настоящего стандарта является ожидаемым поведением.

F.3.5 Отображение двоичных значений

Свойство, определенное ключевым словом BINARY (двоичное), отображается в объект P21.Binary. Метод valueOf() возвращает двоичный литерал без кавычек в начале и в конце строки, представленный символьной строкой ECMAScript. Метод toString() возвращает точно такой же результат, что и метод valueOf(). Метод toP21String() возвращает строку символов с начальными и конечными кавычками на своих местах.

Пример — Выражение ECMAScript "model.example.$value = new P21.Binary ('0123456789ABCDEF');" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = "0123456789ABCDEF";
..
ENDSEC;
```

F.3.6 Отображение имен экземпляров объектов

Свойство, определенное ключевым словом ENTITY_INSTANCE_NAME (имя экземпляра объекта) отображается в объект P21.EID. Метод valueOf() возвращает кодированный объект, на который указывает имя экземпляра объекта или NULL.

Примечание — Для элемента привязки, который определяется объектным типом данных, должно быть дано зависящее от прикладной области отображение ECMAScript.

Метод toString() возвращает кодированное как символьная строка ECMAScript имя экземпляра объекта без начального символа "#". Метод toP21String() возвращает символьную строку, кодированную в соответствии с тем, как описано в настоящем стандарте. Строка включает начальный символ "#".

Пример — Выражение ECMAScript "model.example.$value = new P21.EID ('20');" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = #20;
..
ENDSEC;
```

F.3.7 Отображение имени экземпляра значения

Свойство, определенное ключевым словом VALUE_INSTANCE_NAME (имя экземпляра значения), отображается в объект P21.VID. Метод valueOf() возвращает кодированное значение, на которое ссылается экземпляр переменной с рассматриваемым именем. Метод toString() возвращает кодированное как символьная строка ECMAScript имя экземпляра значения без начального символа "@". Метод toP21String() возвращает символьную строку, кодированную в соответствии с тем, как описано в настоящем стандарте. Строка включает начальный символ "@".

Пример — Выражение ECMAScript "model.example.$value = new P21.VID ('20');" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = @20;
..
ENDSEC;
```

F.3.8 Отображение имени экземпляра константы

Свойство, определенное ключевым словом CONSTANT_ENTITY_NAME (имя постоянного объекта), отображается в объект P21.CIN. Метод valueOf() возвращает кодированный объект, на который указывает имя экземпляра объекта или NULL.

Примечание — Для элемента привязки, определяемого константой EXPRESS, имя константы, возможно, является более полезным, чем значение.

Метод toString() возвращает кодированное как символьная строка ECMAScript имя экземпляра константы без начального символа "#". Метод toP21String() возвращает символьную строку, кодированную в соответствии с тем, как описано в настоящем стандарте. Строка включает начальный символ "#".

Пример — Выражение ECMAScript "model.example.$value = new P21.CIN ('INCH');" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = #INCH;
..
ENDSEC;
```

F.3.9 Отображение имени значения константы

Свойство, определенное ключевым словом CONSTANT_VALUE_NAME (имя значения константы), отображается в объект P21.CVN. Метод valueOf() возвращает кодированный объект, на который указывает имя значения константы.

Метод toString() возвращает кодированное как символьная строка ECMAScript имя экземпляра значения без начального символа "@". Метод toP21String() возвращает символьную строку, кодированную в соответствии с тем, как описано в настоящем стандарте. Строка включает начальный символ "@".

Пример — Выражение ECMAScript "model.example.$value = new P21.CVN ('PI');" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = @PI;
..
ENDSEC;
```

F.3.10 Отображение нулевого значения

Свойство, определенное пустым (не присвоенным) значением ("$"), отображается в значение null (ноль) ECMAScript.

Пример — Выражение ECMAScript "model.example.$value = null;" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = $;
..
ENDSEC;
```

F.3.11 Отображение списка элементов привязки

Свойство, определенное ключевым словом ANCHOR_ITEM_LIST (список элементов привязки), отображается в объект P21.List. Метод valueOf() возвращает массив ECMAScript, в котором каждый член исходного списка представлен с применением кодировки, предписанной для соответствующего метода valueOf(), описанного в F.3. Метод toString() возвращает одномерный массив ECMAScript, в котором каждый член исходного списка представлен с применением кодировки, предписанной для соответствующего метода toString(). Метод toP21String() возвращает одномерный ECMAScript, в котором каждый член исходного списка представлен с применением кодировки, предписанной для соответствующего метода toP21String().

Пример — Выражение ECMAScript "model.example.$value = new P21.List (new P21.Integer (1), new P21.Integer(2), new P21.Integer(3));" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
<example> = (1, 2, 3);
..
ENDSEC;
```

F.3.12 Отображение URI

Свойство, определенное ключевым словом RESOURCE (ресурс), отображается в объект P21.URI.

Если значение, на которое ссылается элемент привязки, является обменной структурой, метод valueOf() возвращает элемент привязки, на который содержится ссылка в указанном ресурсе. Элемент привязки кодируется так, как описано в настоящем приложении.

Если значение, на которое ссылается элемент привязки, не является обменной структурой, метод valueOf() возвращает значение NULL (ноль).

Метод toString() возвращает значение URI, представленное как символьная строка, в которой опущены начальный "<" и завершающий ">" символы. Метод toP21String() возвращает строку символов, в которую включены начальный "<" и завершающий ">" символы.

Пример — Выражение ECMAScript "model.example.$value = new P21.URI ('#wheel');" эквивалентно следующему коду в структуре обмена:

```
ANCHOR;
..
#10 = <#wheel>;
..
ENDSEC;
```

F.4 Требуемые методы объекта P21

Объект P21.Model должен иметь следующие методы:

F.4.1 Метод uri()

Метод uri() возвращает адрес обменной структуры в форме объекта P21.URI.

F.4.2 Метод name()

Метод **name()** возвращает значение атрибута **file_name.name** (наименование файла, наименование) (см. 8.2.3) заголовочной секции в форме объекта P21.String.

F.4.3 Метод schema_population()

Метод schema_population() возвращает массив объектов P21.Population, описывающих состав ссылок, который содержит объект обменной структуры (см. 8.2.5).

Каждый объект P21.Population имеет следующие методы:

- метод uri() возвращает, в форме объекта P21.URI, адрес обменной структуры, содержание которой должно быть включена в число экземпляров объектов;
- метод stamp() возвращает null (ноль) или дату и штамп времени в форме объекта ECMAScript date;
- метод verification() возвращает значение true (истина), если структура, на которую дана ссылка, не менялась с момента задания проверки, иначе метод возвращает значение 'false' (ложь).

F.4.4 Метод set_uri()

Метод set_uri() устанавливает для обменной структуры адрес, задаваемый в форме объекта P21.URI, играющего роль параметра метода.

F.4.5 Метод set_name()

Метод set_name() устанавливает для обменной структуры значение атрибута **name** (имя) объекта **file_name** (имя файла) заголовочной секции, задаваемое в форме объекта P21.String, играющего роль параметра метода.

Примечание — Приложения используют значение атрибута **name** (имя) объекта **file_name** (имя файла) заголовочной секции для того, чтобы определить представление средствами ECMAScript элементов привязки в обменной структуре.

F.4.6 Метод set_schema_population()

Метод set_schema_population() устанавливает набор объектов **schema_population**, задаваемый в форме играющего роль параметра метода массива объектов P21.Population.

Каждый объект P21.Population имеет следующие методы:

- метод set_uri() устанавливает для обменной структуры адрес, задаваемый в форме объекта P21.URI, играющего роль параметра метода;
- метод set_stamp() устанавливает для входа дату и штамп времени, задаваемые в форме объекта ECMAScript date, играющего роль параметра метода;
- метод set_verification() создает и устанавливает дайджест обменной структуры, на которую дается ссылка.

Пример — Следующие строки кода создают новый объект P21.Population и добавляют его в число объектов, образующих набор объектов schema_population:

```
pop = new P21.Population;
pop.set_uri (new P21.URI ('http//gost.e-ecolog.ru                    ));
pop.set_stamp (new date ('4/2/2013'));
pop.set_verification ();
model.set_schema_population (model.schema_population().push (pop));
```

Приложение G
(обязательное)

Отображение UUID в имена элементов привязки

G.1 Введение

Многие системы создают уникальные идентификаторы. Обычно такие идентификаторы называются Универсальные уникальные идентификаторы (UUID) или Глобальные уникальные идентификаторы (GUID). В настоящем приложении описано, как отобразить эти идентификаторы в имена элементов привязки так, чтобы приложения могли определить, когда два элемента привязки имеют одинаковые UUID.

G.2 Отображение UUID в имена элементов привязки

UUID является 128-битным значением, создаваемым системой.

Примечание — Практически идентификаторы уникальны, но уникальность не гарантируется.

Когда UUID используется как имя элемента привязки, UUID должен кодироваться в соответствии с предписаниями из RFC 4122 (см. 3.1.7.4). При кодировании не должен включаться префикс.

Примечание — Кодировка RFC 4122 выбрана вследствие того, что использование тире делает коды чрезвычайно легкими для распознавания человеком случаев, когда два идентификатора UUID имеют одинаковые значения и, следовательно, представляют один и тот же идентификатор.

Пример

Правильное представление имени элемента привязки	Источник
<48a0de4c-3c6f-488f-843a-231e08125315>	Создано онлайн-генератором UUID gost.e-ecolog.ru gost.e-ecolog.ru
<63309550-ce63-11e4-8830-0800200c9a66>	Создано онлайн-генератором UUID gost.e-ecolog.ru gost.e-ecolog.ru

Неправильное представление	Проблема
<439K6>	Неверная кодировка RFC 4122.
<uuid:09087c40-ce64-11e4-8830-0800200c9a66>	Код не должен содержать префикс.

Приложение Н
(справочное)

Пример полной структуры обмена

Н.1 Введение

Ниже представлен пример определения средствами EXPRESS-схемы, таблицы сокращенных имен и структуры обмена. Приведенная EXPRESS-схема не отражает содержания какого-либо стандарта серии ИСО 10303.

Н.2 Пример схемы

Определение EXPRESS-схемы. используемые в примере структуры обмена.

```
SCHEMA example_geometry;

TYPE length_measure = NUMBER;
END_TYPE;

ENTITY geometry
SUPERTYPE OF (ONEOF(point));
END_ENTITY;

ENTITY point
SUPERTYPE OF (ONEOF(cartesian_point))
SUBTYPE OF (geometry);
END_ENTITY;

ENTITY cartesian_point
SUBTYPE OF (point);
  x_coordinate  : length_measure;
  y_coordinate  : length_measure;
  z_coordinate  : OPTIONAL length_measure;
END_ENTITY;

TYPE edge_or_logical = SELECT (edge, edge_logical_structure);
END_TYPE;

ENTITY topology
SUPERTYPE OF (ONEOF(vertex, edge, loop));
END_ENTITY;

ENTITY vertex
SUBTYPE OF (topology);
  vertex_point : OPTIONAL point;
END_ENTITY;

ENTITY edge
SUBTYPE OF (topology);
  edge_start   : vertex;
  edge_end     : vertex;
END_ENTITY;

ENTITY edge_logical_structure;
  edge_element : edge;
  flag : BOOLEAN;
END_ENTITY;

ENTITY loop
SUPERTYPE OF (ONEOF(edge_loop))
SUBTYPE OF (topology);
END_ENTITY;
```

```
ENTITY edge_loop
SUBTYPE OF (loop);
  loop_edges : LIST [1:?] OF edge_or_logical;
END_ENTITY;

END_SCHEMA;
```

Н.3 Пример сокращенных имен

Ниже даны сокращенные имена объектов вышеприведенной схемы.

Имя объекта	Сокращенное имя
cartesian_point	cpt
vertex	vx
edge	ed
edge_logical_structure	ed_strc

Н.4 Пример структуры обмена

Ниже приведен пример полной структуры обмена.

Примечание — Поскольку схема является только примером, в заголовочной секции экземпляра объекта **file_schema** в атрибуте **schema_name** отсутствует регистрационный идентификатор схемы (id).

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('THIS FILE CONTAINS A SMALL SAMPLE STEP MODEL'),'3;1');
FILE_NAME('EXAMPLE STEP FILE #1',
'2013-02-11T15:30:00',
('JOHN DOE',
'ACME INC.',
'METROPOLIS USA'),
('ACME INC. A SUBSIDIARY OF GIANT INDUSTRIES','METROPOLIS USA'),
'CIM/STEP VERSION2',
'SUPER CIM SYSTEM RELEASE 4.0',
'APPROVED BY JOE BLOGGS');
FILE_SCHEMA(('EXAMPLE_GEOMETRY'));
ENDSEC;
DATA;
/*
   СЛЕДУЮЩИЕ 13 ОБЪЕКТОВ ПРЕДСТАВЛЯЮТ КОНТУР СТОРОН ТРЕУГОЛЬНИКА
*/
#1=CPT(0.0,0.0,0.0);    /* ЭТО ОБЪЕКТ ТИПА ДЕКАРТОВА ТОЧКА */
#2=CPT(0.0,1.0,0.0);
#3=CPT(1.0,0.0,0.0);
#11=VX(#1);             /* ЭТО ОБЪЕКТ ТИПА ВЕРШИНА */
#12=VX(#2);
#13=VX(#3);
#16=ED(#11,#12);        /* ЭТО ОБЪЕКТ ТИПА СТОРОНА */
#17=ED(#11,#13);
#18=ED(#13,#12);
#21=ED_STRC(#17,.F.);   /* ЭТО ОБЪЕКТ ТИПА ЛОГИЧЕСКАЯ СТРУКТУРА СТОРОНЫ */
#22=ED_STRC(#18,.F.);
#23=ED_STRC(#16,.T.);
#24=ED_LOOP((#21,#22,#23));   /* ЭТО ОБЪЕКТ ТИПА КОНТУР СТОРОН */
/*
   ВОЗМОЖНЫ ДРУГИЕ СИНТАКСИЧЕСКИЕ ПРЕДСТАВЛЕНИЯ. ПРЕДЫДУЩИЙ ПРИМЕР
   ПРЕДСТАВЛЯЕТ ОДИН ИЗ ВОЗМОЖНЫХ МЕТОДОВ.
*/
ENDSEC;
END-ISO-10303-21;
```

Примечание — Данный пример структуры обмена был отредактирован для улучшения его читаемости, для чего добавлены пробелы, не являющиеся обязательными.

Приложение I
(справочное)

Пример распределенной структуры обмена

I.1 Введение

Настоящее издание позволяет распределять содержимое обменной структуры между несколькими файлами. Распределение не меняет смысла содержимого, а только положение. В примере ниже показано распределение данных из приложения H.4 между двумя файлами.

I.2 Пример распределенной обменной структуры

В следующем примере предполагается, что был принят файл компании Giant Industries и в новых условиях Acme Inc., подразделение компании определяет вершины по точкам, предоставленным компанией Giant Industries.

Компания Giant включила дайджест сообщения для файла среди экземпляров объекта схемы (см. 8.2.5). Файлы Giant имеют подпись, которая подтверждает как содержимое этих файлов, так и содержимое файла Acme вследствие наличия дайджеста сообщения.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('ЭТОТ ФАЙЛ ПРЕДСТАВЛЯЕТ ТОТ ЖЕ НЕБОЛЬШОЙ ПРИМЕР STEP-МОДЕЛИ, ЧТО И В ПРИЛОЖЕНИИ H '),'4;2');
FILE_NAME('gost.e-ecolog.ru
'2013-02-11T16:30:00',
('JANE BOSS',
'GIANT INDUSTRIES.',
'MEGALOPOLIS USA'),
('GIANT INDUSTRIES','MEGALOPOLIS USA'),
'CIM/STEP VERSION3',
'SUPER CIM SYSTEM RELEASE 5.0',
'APPROVED BY MR. BIG');
FILE_SCHEMA(('EXAMPLE_GEOMETRY'));
SCHEMA_POPULATION((('ftp://ftp.acme.net/second_file.stp',$,'44245c2ff046a5d65be9a33242d8c8c9ba9002d387d8b113dd1516bee735ab60')));
ENDSEC;

ANCHOR;
<POINT_1> = #1;
<POINT_2> = #2;
<POINT_3> = #3;
<POINT_4> = #4;
<POINT_5> = #5;
<POINT_6> = $;        /* ПЕРСПЕКТИВНОЕ ИСПОЛЬЗОВАНИЕ */
ENDSEC;

REFERENCE;
#11 = <ftp://ftp.acme.net/second_file.stp#vertex_1>;
ENDSEC;

DATA;
/*
 СЛЕДУЮЩИЕ ОБЪЕКТЫ ПРЕДСТАВЛЯЮТ ВСЕ, ЗА ИСКЛЮЧЕНИЕМ ОДНОЙ ВЕРШИНЫ РЕБЕР КОНТУРА ТРЕУГОЛЬНИКА
*/
#1=CPT(0.0,0.0,0.0);   /* ДЛЯ ДЕКАРТОВЫХ ТОЧЕК СОЗДАНЫ ЭЛЕМЕНТЫ ПРИВЯЗКИ, ТАК ЧТО НА НИХ МОЖНО ДЕЛАТЬ ССЫЛКИ ИЗВНЕ*/
#2=CPT(0.0,1.0,0.0);
#3=CPT(1.0,0.0,0.0);
#4=CPT(0.0,2.0,0.0);     /* ДЛЯ ТОГО, ЧТОБЫ ДАТЬ КОМПАНИИ ACME БОЛЬШЕ ВОЗМОЖНОСТЕЙ, СОЗДАНЫ НОВЫЕ ТОЧКИ */
#5=CPT(2.0,0.0,0.0);
```

```
#12=VX(#2);            /* ОДНА ВЕРШИНА БЫЛА ПЕРЕМЕЩЕНА В ДРУГОЙ ФАЙЛ*/
#13=VX(#3);
#16=ED(#11,#12);       /* ДАННЫЕ РЕБРА НЕ ИЗМЕНЕНЫ, ОБЪЕКТ С ИМЕНЕМ #11 ОПРЕДЕЛЕН В СЕКЦИИ ССЫЛОК*/
#17=ED(#11,#13);
#18=ED(#13,#12);
#21=ED_STRC(#17,.F.);    /* ОБЪЕКТ, ПРЕДСТАВЛЯЮЩИЙ ЛОГИЧЕСКУЮ СТРУКТУРУ РЕБРА ТАКЖЕ НЕ ИЗМЕНЯЛСЯ*/
#22=ED_STRC(#18,.F.);
#23=ED_STRC(#16,.T.);
#24=ED_LOOP((#21,#22,#23));   /* ОБЪЕКТ, ПРЕДСТАВЛЯЮЩИЙ КОНТУР РЕБЕР, ТАКЖЕ НЕ ИЗМЕНЯЛСЯ*/

   ВОЗМОЖНЫ ДРУГИЕ РАСПРЕДЕЛЕНИЯ. РАСПРЕДЕЛЕНИЕ ТОЛЬКО ОДНОГО
   ОБЪЕКТА В ПРАКТИЧЕСКОЙ ДЕЯТЕЛЬНОСТИ НЕВЕРОЯТНО И МОЖЕТ ПРИВЕСТИ
   К НИЗКОЙ ПРОИЗВОДИТЕЛЬНОСТИ
*/
ENDSEC;
END-ISO-10303-21;
SIGNATURE
A1yBCCQAc27kxxdf3iMQTxg+4jKqYRN6TPnHmV3ZQfyFwmj5Bf76SkvHx0DnJN3O
fpzh2x7n4Ui+nxuu7JeuP3YYNWj4Qo8Etn/3/26nRKdM3tTWapUo3F7Ul5GPOEi+
uZ/jYNyagLwvulNFM5sqUdI01Nx6C38O1NTUsbCpIZ39X/M2i7DBNQQ72qxWCiWW
JJfCygnf9TwdlAMR+WzBzb4qzUH682wWyeCU5TgYYLY1XFcUrM2Wts0Y3yGvXSLI
uZGoEQNdblctS0ogEub2nPXyJDAbH337gCvljQPwBld/xGU4hgwZE4dSIRc51kGH
...
ENDSEC;
```

Файл подразделения Acme содержит только один объект.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('В НАСТОЯЩЕМ ФАЙЛЕ ПРЕДСТАВЛЕН ПОДРАЗДЕЛ МОДЕЛИ STEP ИЗ ПЕРВОГО ФАЙЛА›),›4;2›);
FILE_NAME('ftp.acme.net/second_file.stp',
'2013-02-11T17:30:00',
('JOHN DOE',
'ACME INC.',
'METROPOLIS USA'),
('ACME INC. A SUBSIDIARY OF GIANT INDUSTRIES','METROPOLIS USA'),
'CIM/STEP VERSION2',
'SUPER CIM SYSTEM RELEASE 4.0',
'APPROVED BY JOE WILLING');
FILE_SCHEMA(('EXAMPLE_GEOMETRY'));/* НЕЯВНО ПРЕДСТАВЛЕННАЯ СХЕМА */
ENDSEC;

ANCHOR;
<vertex_1> = #11;
ENDSEC;

REFERENCE;
#1 = <gost.e-ecolog.ru                                    >; /* ОБРАТНАЯ ССЫЛКА НА ПЕРВЫЙ ФАЙЛ*/
ENDSEC;

DATA;
/*
   СЛЕДУЮЩИЕ ОБЪЕКТЫ ПЕРЕМЕЩЕНЫ В НАСТОЯЩИЙ ФАЙЛ
*/
#11=VX(#1);
ENDSEC;
END-ISO-10303-21;
```

Примечание — Для читаемости примера имена экземпляров объектов те же самые, что и в примере Н.4, приложение Н. Этого не требуется, и на практике такое маловероятно. Добавлены не являющиеся необходимыми пробелы и переводы строк.

Приложение J
(справочное)

Примеры констант EXPRESS для определения единиц измерения

J.1 Константы EXPRESS для основных единиц измерения

В следующем примере показано, как определить основные единицы измерения, используемые в обменном файле STEP в качестве констант на языке EXPRESS.

```
CONSTANT
plane_angle_radian : si_unit := named_unit (?) || plane_angle_unit () || si_unit (?, radian);
solid_angle_steradian : si_unit := named_unit (?) || solid_angle_unit () || si_unit (?, steradian);
length_millimeter : si_unit := named_unit (?) || length_unit () || si_unit (milli, metre);
END_CONSTANT;

CONSTANT
seven_dimensions_a : dimensional_exponents := dimensional_exponents (0., 0., 0., 0., 0., 0., 0.);
degree_to_radian : measure_with_unit := measure_with_unit (plane_angle_measure (0.01745329252), radian));
plane_angle_degree : conversion_based_unit := conversion_based_unit ('degree', degree_to_radian))
                                    || named_unit (seven_dimensions_a)
                                                    || plane_angle_unit ();
END_CONSTANT;

CONSTANT
seven_dimensions_b : dimensional_exponents := dimensional_exponents (1., 0., 0., 0., 0., 0., 0.);
inch_to_millimeter : measure_with_unit := measure_with_unit (length_measure (25.4), inch));
imperial_length_inch : conversion_based_unit := conversion_based_unit ('inch', inch_to_millimetre))
                                    || named_unit (seven_dimensions_b)
                                                    || length_unit ();
END_CONSTANT;
```

Примечание — Эти примеры служат только в качестве иллюстраций.

J.2 Использование констант EXPRESS в обменной структуре

В следующем примере для определения геометрического контекста в обменном файле STEP используются предварительно определенные константы EXPRESS.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(
/* description */ ('example of how to reference EXPRESS constants that define units'),
/* implementation_level */ '4;3');

FILE_NAME(
/* name */ 'example.stp',
/* time_stamp */ '2013-02-09T12:37:49-05:00',
/* author */ (''),
/* organization */ (''),
/* preprocessor_version */ '',
/* originating_system */ '',
/* authorisation */ '');
FILE_SCHEMA(('AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF'));  /*
AP242 Schema */

ENDSEC;

DATA;
#10 = (
GEOMETRIC_REPRESENTATION_CONTEXT(3)
GLOBAL_UNIT_ASSIGNED_CONTEXT((#IMPERIAL_LENGTH_INCH, #PLANE_ANGLE_RADIAN, #SOLID_ANGLE_
STERADIAN)
REPRESENTATION_CONTEXT('','3D')
);
....
ENDSEC;

END-ISO-10303-21;
```

Приложение К
(справочное)

Рекомендуемые имена типов файлов

Согласно резолюции 583 (Штутгарт, Германия, июнь 2003), 4-й подкомитет 184 Технического комитета ИСО рекомендует следующие имена типов файлов (расширения) для файлов, описанных в настоящем стандарте:

- имена типов .stp и .step для файлов, соответствующих прикладным протоколам ИСО 10303;
- имя типа .p21 для общего использования, включая файлы стандартов ИСО 10303, ИСО 13584 и ИСО 15926.

Приложение L
(справочное)

Руководство по распечатке структуры обмена

L.1 Общие положения

Структура обмена допускает включение необязательных директив, которые явно управляют представлением структуры в напечатанном виде. Когда такие директивы не включены и должно быть создано печатное представление структуры обмена, следует использовать набор неявных указаний по управлению процессом печати, определенный в G.2. Явные директивы управления при печати перекрывают неявные указания по управлению процессом печати.

L.2 Явные директивы управления процессом печати

Явные директивы управления процессом печати могут быть использованы в случаях, когда отправитель нуждается в точном управлении появлением структуры обмена в напечатанном виде.

Директива *обратная косая черта, прописная буква N, обратная косая черта* "\N\" указывает, что первый символ, следующий сразу за директивой, появляется в начале новой строки. Директива *обратная косая черта, прописная буква F, обратная косая черта* “\F\” указывает, что печать будет продолжена с новой страницы. В любом случае директивы управления процессом печати сами в напечатанном виде не появляются.

Примечание — Процесс печати структуры обмена обычно не должен управляться явными или неявными директивами управления при печати. Подразумевается, что директивы управления процессом печати будут использоваться только тогда, когда отправитель требует управления представлением структуры обмена в напечатанном виде. Эта ситуация может иметь место, если структура обмена является частью официального контракта.

L.3 Неявные директивы управления процессом печати

При отсутствии явных директив управления при печати структуры обмена могут быть использованы следующие директивы:

a) строка выключается слева;

b) каждая секция начинается с новой строки;

c) объекты заголовочной секции должны начинаться с начала новой строки;

d) имя экземпляра объекта, предшествующее *знаку равенства* "=", должно начинаться с новой строки;

e) лексемы, не являющиеся строками (string) и двоичными значениями (binary), не должны разделяться между строками. Строки и двоичные значения могут быть разделены только в том случае, если они целиком не умещаются в одной строке. Директива h) определяет, где должны разделяться строки и двоичные значения;

f) комментарии должны начинаться с новой строки;

g) разделитель лексем, не являющийся комментарием, не должен разделяться между строками. Комментарий может разделяться, если он не помещается в одной строке. Директива h) определяет, где будут разделяться комментарии;

h) строки должны иметь длину не больше, чем 72 символа. Если символ не может быть напечатан в качестве 73-й позиции, его следует напечатать в первой позиции новой строки.

Приложение М
(справочное)

Журнал изменений

М.1 Изменения, внесенные во второе издание:

добавлен пункт 8.2.6 для описания **file_population**;

добавлен пункт 8.2.7 для описания **section_language**;

добавлен пункт 8.2.8 для описания **section_context**;

расширен пункт 12.1.7, позволяющий использовать краткие имена для перечисляемых типов;

расширен пункт 11.1, позволяющий использовать несколько секций данных;

добавлено обязательное приложение Е для описания проверки структур обмена с несколькими секциями данных.

М.2 Изменения, внесенные в третье издание:

добавлена секция привязки для описания того, как объектам в структуре обмена могут быть присвоены имена привязки, на которые можно ссылаться из внешних файлов в раздел 9;

добавлена ссылочная секция для описания того, как можно ссылаться на экземпляры объектов, хранящиеся во внешних файлах, из структуры обмена в раздел 10;

добавлена секция подписи для обеспечения подлинности структуры обмена в раздел 14;

добавлено в пункт 8.2.5 определение заполнения схемы в заголовочной секции, чтобы описать, как найти распределенное заполнение для проверки соответствия;

добавлены три новых уровня реализации **file_description** заголовочной секции в пункте 8.2.2;

изменен подраздел 4.3 для определения трех уровней синтаксического соответствия;

изменен подраздел 5.3, чтобы исключить требование, чтобы структура обмена содержала хотя бы одну секцию данных;

изменен пункт 6.4.4, чтобы разрешить экземпляры значений и ссылки на константы EXPRESS;

изменен раздел 11, чтобы разрешить константы EXPRESS и экземпляры значений;

добавлен пункт А.4 для передачи сжатого содержимого;

добавлено нормативное приложение F для управления привязками ECMAScript;

добавлено нормативное приложение G для кодирования UUID в именах привязки;

добавлен информативный пример распределенной модели в приложении I;

добавлены информативные примеры определений единиц измерения и констант в приложении J;

добавлены рекомендации по названию типа файла в приложении К;

изменен 5.2, чтобы описать основной алфавит в терминах ИСО/МЭК 10646, а не ИСО/МЭК 8859-1, и указать кодировку UTF-8 для всех символов. Приложение D в издании 2, в котором указан графический символ для каждого значения октета, было удалено;

расширенная строковая кодировка в 6.4.3 позволяет кодовым точкам от U + 0080 до U + 10FFFF отображаться в строках как UTF-8 кодированные октеты, как альтернатива более ранним кодировкам \X2\ и \X4\. Переработан раздел для использования текущей терминологии ИСО/МЭК 10646;

уточнена последовательность сортировки для значений частичных сложных объектов во внешнем отображении в 12.2.5.3.

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным стандартам

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ISO 639-2	—	*
ISO 8601:2004	—	*
ISO 10303-1:1994	IDT	ГОСТ Р ИСО 10303-1—99[1)] «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1. Общие представления и основополагающие принципы»
ISO 10303-11:2004	IDT	ГОСТ Р ИСО 10303-11—2009 «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 11. Методы описания. Справочное руководство по языку EXPRESS»
ISO/IEC 8824-1	IDT	ГОСТ Р ИСО/МЭК 8824-1—2001 «Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации»
ISO/IEC 8859-1	—	*
ISO/IEC 8859 (all parts)	—	*
ISO/IEC 10646	—	*
ISO/IEC 16262	—	*
ISO/IEC 21320-1	—	*

* Соответствующий национальный стандарт отсутствует. До его принятия рекомендуется использовать перевод на русский язык данного международного стандарта.

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:

- IDT — идентичные стандарты.

[1)] Действует ГОСТ Р ИСО 10303-1—2022, идентичный ИСО 10303-1—2021.

Библиография

[1] ISO/IEC 3788, Information processing — 9-Track, 12,7 mm (0.5 in) wide magnetic tape for information interchange using phase encoding at 126 ftpmm (3200 ftpi), 63 cpmm (1600 cpi)

[2] Globally Unique IDentifier (GUID): 2.5.5 Globally Unique Identifiers (GUIDs), [cited 2013-01-17], Available from World Wide Web, gost.e-ecolog.ru .

[3] Representational state transfer (REST): How to create a REST Protocol, O'Reilly XML.com, 1 December 2004, [cited 2013-01-17]. Available from World Wide Web, gost.e-ecolog.ru

[4] Cryptographic Message Syntax (CMS): How to digitally sign, digest, authenticate, or encrypt arbitrary message content, Internet Engineering Task Force RFC 5652 September 2009 [cited 2015-03-10]. Available from World Wide Web:< gost.e-ecolog.ru >

[5] Extensible Markup Language (XML): How to enable generic SGML to be served, received, and processed on the Web, 1.0 (Fith Edition) World Wide Web Consortium Recommendation 26 November 2008 [cited 2013-03-15]. Available from World Wide Web:< gost.e-ecolog.ru >

[6] The Base15, Base32 and Base64 Data Encodings: How to use base 64, base 32, and base 16 encoding schemes, Internet Engineering Task Force RFC 4648 October 2006 [cited 2015-03-11]. Available from World Wide Web:< http:// gost.e-ecolog.ru >

[7] Uniform Resource Identifiers (URI): How to use and define functionality for a URI, Internet Engineering Task Force RFC 3986 January 2005 [cited 2015-03-09]. Available from World Wide Web:< gost.e-ecolog.ru >

[8] A Universally Unique IDentifier (UUID): How to define a Uniform Resource Name namespace for UUIDs (Universally Unique IDentifier), also known as GUIDs (Globally Unique IDentifier), Internet Engineering Task Force RFC 4122 July 2005 [cited 2015-03-19]. Available from World Wide Web:< gost.e-ecolog.ru >

УДК 656.072:681.3:006.354 ОКС 25.040.40

Ключевые слова: прикладные автоматизированные системы, промышленные изделия, представление данных, обмен данными, элемент представления, контекст представления

Редактор *Л.В. Коретникова*
Технический редактор *В.Н. Прусакова*
Корректор *О.В. Лазарева*
Компьютерная верстка *А.Н. Золотаревой*

Сдано в набор 24.11.2022. Подписано в печать 02.12.2022. Формат 60×84⅛. Гарнитура Ариал.
Усл. печ. л. 10,23. Уч.-изд. л. 9,25.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Создано в единичном исполнении в ФГБУ «Институт стандартизации» для комплектования Федерального информационного фонда стандартов, 117418 Москва, Нахимовский пр-т, д. 31, к. 2.

gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru